Speech/Music Discrimination: Novel Features in Time Domain

ALNADABI, MUHAMMAD,SAEID,MUHAMMAD

Speech/Music Discrimination:

Novel Features in Time Domain

Muhammad Alnadabi

Degree of Doctor of Philosophy

University of Durham

April 2010

Dedication

To the souls of my mother, my father, and my younger brother

To my sisters, my wife, and my sons: Saeid, Abdullah, and Ammar

To my dearest friends who supported me with their prayer and advice

I cannot reward you for your support, but I can only wish you the best of all your wishes and what leads you to the true eternal happiness.

Abstract

This research aimed to find novel features that can be used to discriminate between speech and music in the time domain for the purpose of data retrieval. The study used speech and music data that were recorded in standard anechoic chambers and sampled at 44.1 kHz.

Two types of new features were found and thoroughly examined: the Ratio of Silent Frames (RSF) feature and the Time Series Events (TSE) set of features. The Receiver Operating Characteristics (ROC) curves were used to assess each one of the proposed features as well as certain relevant features from the literature for the purpose of comparison. The RSF feature introduced up to 8% enhancement when compared to a couple of relevant features from the literature. One of the TSE set of features provided close to 100% speech/music discrimination.

Acknowledgement

First of all, I would like to thank my supervisor, Dr. Sherri Johnstone, who helped me walk through the steps of PhD until I achieved the write up of this thesis. While my colleagues in other departments complained of having difficulty in meeting with their supervisors, my supervisor was always available.

Secondly, I would like to thank my friends for their prayer for me to succeed and for their "you can do it" spirit throughout my study.

Last, but not least, many thanks and appreciation for my sponsor, Sultan Qaboos University, for the continuous financial support and for being there for me at difficult times.

Table of Contents

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

List of Tables

Chapter 2

Chapter 4

Chapter 5

Chapter 6

Nomenclature

List of symbols/abbreviations used in this thesis in alphabetical order

Abbreviation	Stands for	Units (If Applicable)
AMDF	Average Magnitude Difference Function	
ASR	Automatic Speech Recognition	
DFT	Discrete Fourier Transform	
E	Energy of Signal	Joule (J)
FP	False Positive	
IDFT	Inverse Discrete Fourier Transform	
LEF	Percentage of Low Energy Frames	%
LSTER	Low Short-Time Energy Ratio	
MLER	Modified Low Energy Ratio	
NPR	Non Pitch Ratio	
NSR	Non Silence Ratio	
NZCE	Non Zero Crossing Event	
P(NZCE)	Probability of Non Zero Crossing Event	

Abbreviation	Stands for	Units (If Applicable)
P(ZCE)	Probability Non Zero Crossing Event	
PDF	Probability Density Function	
PMF	Probability Mass Function	
PSTD	Pitch Standard Deviation	
RMS	Root Mean Square	Volts (V)
ROC	Receiver Operating Characteristics	
RSF	Ratio of Silent Frames	
RT	Reverberation Time	Seconds (s)
SC	Spectral Centroid	Hertz (Hz)
SF	Spectral Flux	Hertz (Hz)
SIFT	Simplified Inverse Filter Tracking	
SMD	Speech/Music Discrimination	
SNR	Signal to Noise Ratio	
SPL	Sound Pressure Level	Pascal (Pa), Newton/meter^2 (N/m^2)
SPR	Similar Pitch Ratio	
SR	Spectral Roll-off Point	
TP	True Positive	

Abbreviation	Stands for	Units (If Applicable)
TSE	Time Series Events	
ZC	Zero Crossing	
ZCE	Zero Crossing Event	
ZCR	Zero Crossing Rate	crossings/sample, crossings/second

Chapter 1: Introduction

1.1. Human Auditory System

Discriminating between different types of audio signals is an easy task for the human auditory system. For the first moment one hears a girl screaming he can recognize that from the sound of an ambulance passing by. However, this task is considered to be not as trivial for machines. Yet, no real-time algorithm exists that can replicate the human auditory system. The latter has the capability to carry out many functions that can be broadly categorized into: discrimination and separation. The latter function aims to separate mixed sounds from each other [1-10], while the former one aims to discriminate between the various types of sounds [11-24].

This thesis documents novel audio features that can be used for speech/music discrimination. Chapter 4 explains the first feature called Ratio of Silent Frames (RSF) and experimentally examines its performance. Chapters 5 and 6 explain the second feature called Non Zero Crossing Event (NZCE) and also examines its performance experimentally. The thesis also provides a comparison between the proposed features and some relevant features that exist in the literature.

The discrimination problem can take many hierarchical levels, the simplest being detection of silence [25] where no sound is present. A more advanced hierarchical discrimination method is illustrated by figure 1.1 [1] where speech is classified into male, female, or sport announcer and music is further classified into different genre [26-29]. Bi-discriminators are systems that classify audio signals into one of two types (e.g. noise/speech, silence/non-silence).

Speech/music discrimination (SMD) is one such system where audio signals are classified to be either speech or music and is the main subject of this study.

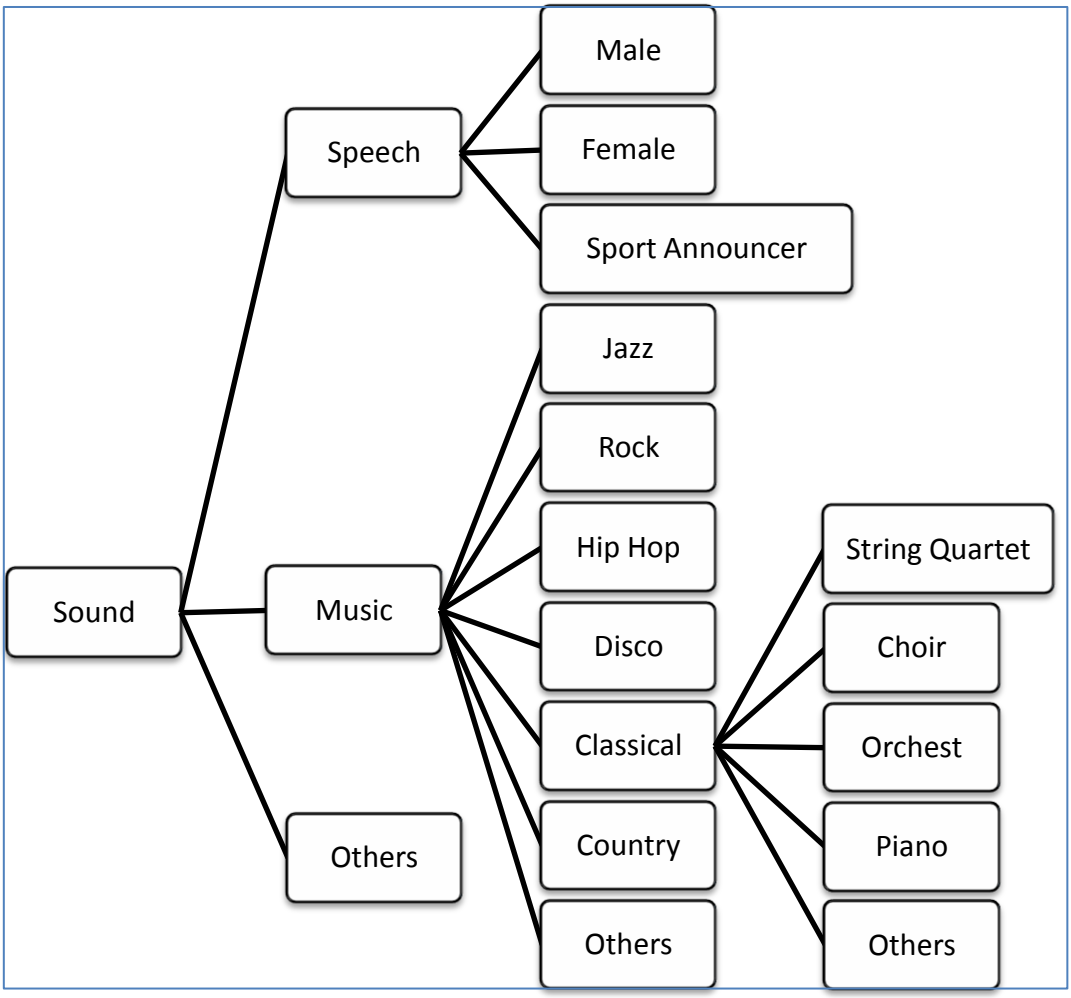


Figure 1.1 Hierarchical classification of different types of sound

1.2. About This Thesis

There is a growing interest in speech processing for various applications; audio type classification is a branch in this field of research. At this early stage of development, the audio classification algorithms that exist in the literature still have certain research questions to be answered. Every new research contributes a novel method to classify audio signals into a hierarchy similar to that shown in figure 1.1. Some broad classifications, including this

thesis, work at the first level; i.e. discriminating between speech and music. Every research contributes to the field of classification by either presenting a novel feature or enhancing an existing one. Sound features can be defined as mathematical algorithms which can be implemented, either by software or hardware, to extract useful information from the signal that is not obvious from the raw data. Features are extracted either from the time (temporal) domain or the frequency (spectral) domain. In time domain the signal is analysed with respect to time, while in the frequency domain the signal is analysed with respect to frequency. Spectrograms are sometimes also used to extract features that carry spectral as well as temporal information. This thesis is contributing to the time domain features.

Speech signals are widely known in the literature of speech/music discrimination to carry more silence (or low energy) frames than music. Such a fact can be sometimes realized when breaking any word into its syllables; e.g. the word "distribution" can be broken into "dist-rib-u-tion". This fact has been exploited in the literature. However, every algorithm in the literature suggested a different method to detect such silence. The first novel feature presented in this thesis is called the Ratio of Silent Frames (RSF) where a new algorithm is proposed by means of a different method of silence detection.

The percentage of Low Energy Frames (LEF) [21, 23, 30] in an audio track is a basic feature that is used in the literature to discriminate between speech and music. The proposed RSF feature was compared to the LEF feature and found to provide an enhancement of up to 8%.

More than two decades ago, Kedem Benjamin [16] introduced the Zero Crossing Rate (ZCR) as a measure of computing the dominant frequency of a signal in the time domain. He suggested that the ZCR is a feature that can be exploited to discriminate between speech and music. Since then, researchers have been investigating the usage of this feature for various applications [31, 32]. However, the zero crossing events (ZCE) are not the only possible

events that could occur in a signal. There are 8 other possible events that have not been investigated yet, totalling 9 types of events. This thesis provides detailed investigation of these 9 events that could occur in a digital audio signal. After a thorough investigation of these events, the Non Zero Crossing Events (NZCE) features are found to provide close to zero misclassification. The thesis provides a comparison between the ZCE features and the NZCE features in chapters 5 and 6, and shows the clear superiority of the NZCE features over the ZCE features.

In summary, this thesis provides the following main contributions to the field of speech/music discrimination:

1. Introducing two novel features that can be used to discriminate between speech and music:
    a. The Ratio of Silent Frames (RSF) feature
    b. The Probability of Non Zero Crossing Events, P(NZCE) features
2. Applying the Receiver Operating Characteristics (ROC) curves to assess any speech/music discrimination feature; applying it to the proposed features as well as certain repeated feature from the literature
3. Thorough analysis of the effect of track length, noise, music genre, and sampling frequency when using the P(NZCE) features

1.3. Motivations and Applications

There are a number of possible applications that might result from being able to label an audio signal as either speech or music. In this section, some of them are described in order for the reader to appreciate the importance of this field of study.

1.3.1. Hearing Aids

One of the most promising applications for the speech/music discrimination is hearing aids. An example of a hearing aid device is demonstrated in figure 1.2. For the user of any hearing aid it is very disturbing when suddenly some loud music is played causing most of the time annoying sounds due to the loudness and the distortion. The latter is mainly due to clipping in the signal as well as due to the amplifier frequency response that is normally designed to operate in the range of speech signals. Music hearing aids do exist, but are more costly; i.e. ranging from $500 to $3000 [33].

The difference in design between speech and music hearing aids is due to two parameters: wider spectrum of music versus speech and differing overall intensity. In the audiology world of research, the term Decibels Hearing Level dBHL is used to refer to the sound intensity. 0 dBHL is the lowest level a normal person can hear. While speech intensity normally ranges from 30 to 35 dBHL; music has a wider range. The latter varies from 20 dBHL for the brushes of a jazz drummer to 120 dBHL for an electric guitar. Sounds beyond 85 dBHL could lead to hearing loss [34].

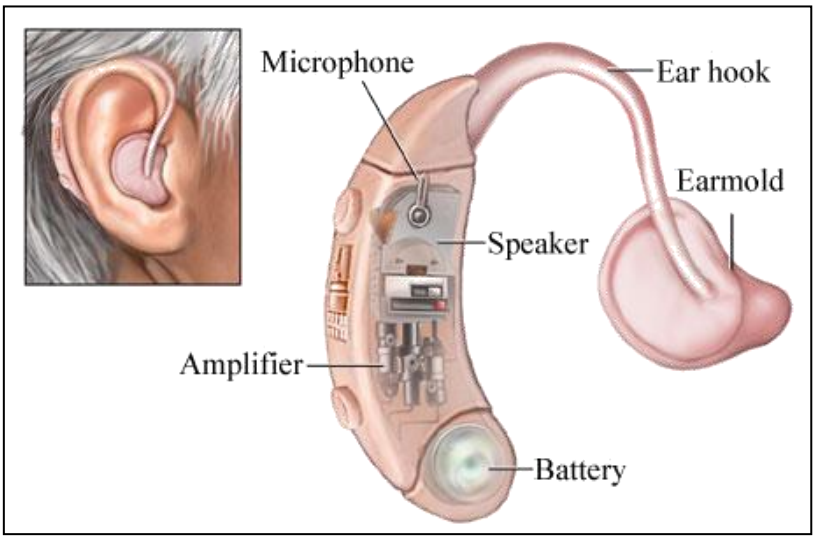


Figure 1.2 An example of a hearing aid

The amplifying circuitry in a speech-hearing aid is normally “oblivious” to these differences and amplifies all picked audio signals equally. Hearing aid users are usually advised to turn the volume of their music players down while turning the volume of their hearing aid up in order to reduce the distortion due to clipping effect; but this would cause a problem when a person is talking to them while listening to music as the speech will not be clearly audible to them. Alternatively, some users simply turn off their hearing aids if they have no control on the music, e.g. in a party or in a shopping area. The way hearing aids behave could be changed depending on the environment and the type of sounds in the vicinity of the user [35].

For this particular application, real time speech/music discrimination is preferred. There is no specific maximum latency as the term “real time” cannot be specifically defined. However, as a rule of thumb, the lower the latency the better the hearing aid would be. In general, 10 seconds latency is too long and 10 milliseconds is non-realistic. It is worth mentioning here that no study has shown latency below one second, including this study. However, the issue of real-time is related to quality. In other words, latency below one second is achievable, but at the cost of higher misclassification error; i.e. it’s a trade-off between delay and quality of discrimination. The details of such trade-off will be thoroughly discussed throughout the rest of the thesis.

1.3.2. Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) systems are designed to convert any recognized speech signal into written text. In any ASR system a stream of audio input could be passed through a pre-processing block to mark each audio segment as either speech (to be translated into text) or music (to be rejected). Figure 1.3 illustrates this concept using a block diagram.

If a musical signal is passed to an ASR system, while being oblivious to its type, it could lead to the wrong translation; i.e. translating the processed sound to the nearest matching text. Furthermore, the time wasted in searching for the word corresponding to a music segment could significantly affect the speed of the ASR system. Thus the performance of an ASR system could be enhanced by adding a speech/music discrimination routine prior to the word recognition process.

The attempt in [20] achieved 91.8% recognition rate after using a speech/music discrimination method to identify non-speech signals and excluding them from being processed by the ASR system. Regardless of the method used, the ASR's performance was improved as it was verified in [20].

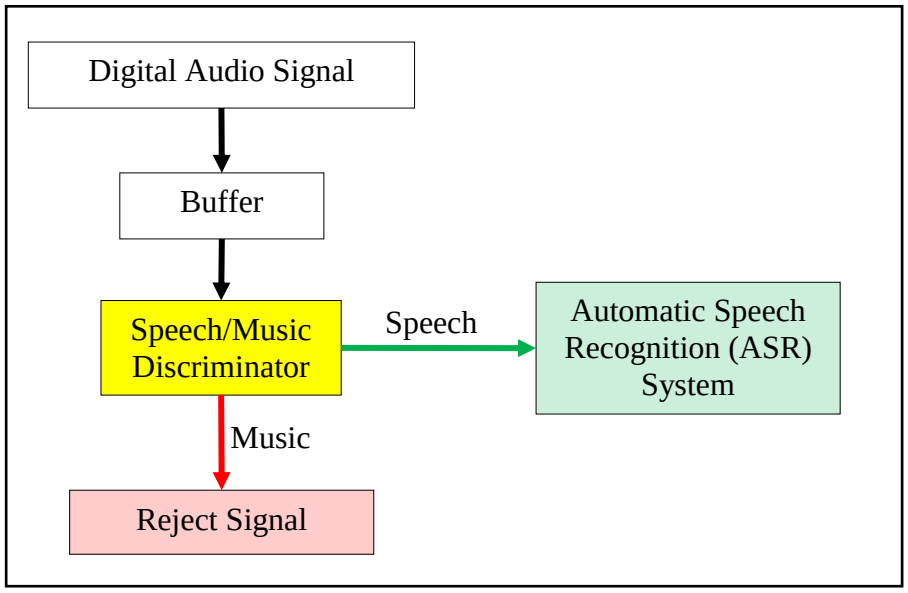


Figure 1.3 Speech/music discriminator being embedded in an Automatic Speech Recognition (ASR) system

1.3.3. Automatic Radio Receiver Scanner

Another application is related to radio receivers. While scanning through radio channels in order to listen to music (or news), the scanner would pause at every channel it detects in order to allow the user to accept or reject the channel after listening to it for a short while. In a highly populated area, with tens of radio channels, this could be a time consuming and frustrating work as one will have to pause to listen to every broadcasted channel before homing into a preferred (e.g. news) channel. The time consumed until one finds his/her favourite type of channel could be reduced by automatically skipping the channel which is not of the desired type. A smarter radio receiver could be designed to make such a decision based on whether the channel found was broadcasting music or speech [21]. In a digital world of radio channels, this task can be achieved by adding some headers at the channel's address which contains information about the type of audio signal that is being broadcasted. However, in an analogue world, this is not usually the case. Furthermore, while listening to news there are musical breaks in the middle. A speech-hearing-aid user would prefer not to be disturbed by such music, but rather mute the sound until the news is spoken again.

1.3.4. Source-Separation versus Type-Discrimination

When a number of simultaneous audio sources (e.g. speech, music, flying aeroplane, and a ringing telephone) are recorded using one microphone it is difficult to separate one source from another. This is known as the "cocktail party problem". The solution to such a problem has resulted in a field of study called *Blind Source Separation (BSS)* [2-4, 6-9] where a system can be designed to extract the various independent sources from their mixture. A speech-music separation system is expected to isolate speech and music sources which were mixed together.

The work and analysis carried out in this thesis is not related to the separation process; rather it is related to the discrimination process [1]. The latter implies the ability to identify whether the examined signal is speech or music. Hence, it is assumed that the analysed signal is either speech or music. Nevertheless, the two systems can complement each other. Once a speech/music separator is applied to separate the mixed signals and the original sources have been separated from each other into their constituent sources, i.e. speech and music, the output of the separator would have two channels. One channel would carry the speech signal and the other channel would carry the music signal. It is then important to distinguish between the two channels by discriminating between speech and music. In order to achieve this discrimination, each channel should be further processed by a speech/music discriminator in order to identify each type; figure 1.4.

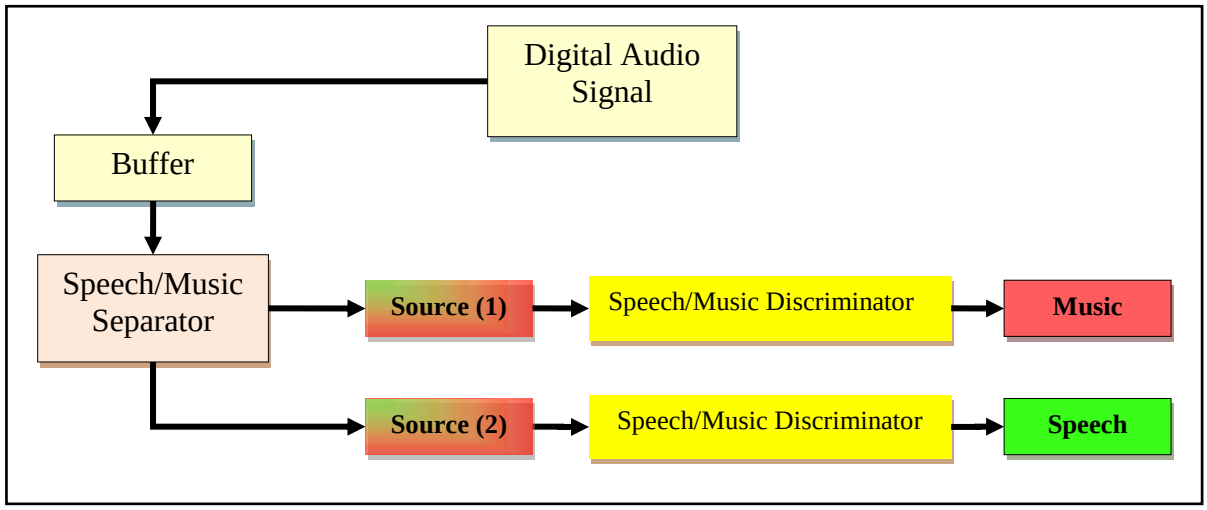


Figure 1.4 Separation of mixed audio sources followed by speech/music discriminators

1.3.5. Video segmentation

Although most video segmentation techniques are based on video analysis [36], audio that accompany any video could also be exploited to assist in video segmentation by identifying the type of audio present in the scene [37]. Some recent work showed that combining video and audio analysis improves the performance of video segmentation [38]. For such applications, discriminating between speech and music adds an advantage in defining scenes of a particular nature. When it comes to understanding the semantic content of a certain video, audio characteristics are equally important, if not more [39]. Moreover, audio-based processing requires far less complex processing than processing video.

1.4. Challenges

Like any project, some challenges were imposed on to this work. Among them was the datasets to be used in the assessment of the proposed features. Since a primary investigation would usually require the elimination of any impurities (e.g. noise, echo) in the original source, the ideal case was to record the data in an anechoic chamber which is designed to have the least possible level of echo and noise. Due to the lack of such a facility in the school of Engineering, an alternative choice was to seek an existing supplier of such data. No single supplier was found to provide both types of audio signals; i.e. speech and music. Instead, every supplier provided a different type. In order for the variation of recording apparatus and environments not to hinder the comparison between speech and music, it was essential to select data suppliers that provided audio recordings with similar specifications.

Ideally, one would like to have two sets which were recorded in exactly the same setup in terms of microphone positioning, chamber dimensions, microphone frequency response, and the distance between the sound source and the microphone. Since this was not feasible, the

least and most important specification that had to be maintained was the sampling rate of the recorded digital audio signals. Also, a certain level of very low noise and echo had to be ensured in both datasets. Speech data from University College London (UCL) and music data from Real World Computing (RWC) had their data recorded at the same sampling rate; i.e. 44.1 kHz. Furthermore, all data were collected in a standard anechoic audio chamber as will be described in chapter 2.

Moreover, it is required to compare the proposed features to some of the features from the literature. As recognized by many authors, such comparison was not possible due to the lack of a unified dataset and a unified assessment method. Every paper used a different dataset and had its own way to assess the performance of the proposed features. Most authors had to repeat what others have done in order to compare it with their feature/method in light of their own framework. Repeating the analysis for all features in the literature is an extremely time consuming work that could not be carried out within the time frame of this research. Nevertheless, it was important to choose certain closely related features to compare with the proposed features.

1.5. Methodology

Every research adopts a different method to reach its objective. In this thesis a very systematic approach was adopted. At first, the various methods in the literature were surveyed in order to make sure that whatever is proposed is novel. Initially, both time domain and frequency domain methods were investigated. The primary objective of the research was to find a method that is of low computational complexity in order to reduce the implementation cost both in terms of money and time. The latter is a requirement for real-time applications. It is known that working in the frequency domain would imply more

complex algorithms than working in the time domain. Thus, it was decided to pursue the study in the time domain. After finding a research gap that needed to be filled the appropriate data were acquired from specialized suppliers.

Computer codes were scripted to analyse the novel methods as well as the ones from the literature. Although analysing every relevant feature involves scripting many pages of codes as shown in Appendix A, this was necessary to reach a fair conclusion about the comparison between the novel features and the relevant ones from the literature. Parameters that are of a significant role to the novel proposed methods were also examined in order to reach a conclusion about the effect of their variation.

1.6. Thesis Structure

This chapter is followed by six more chapters as briefly described below.

- ***Chapter 2: Research Requirements:***

  This chapter details the data used in the analysis throughout the thesis, the environment where the data were recorded, the equipment used in recording the data, and the programming tool utilized to analyse the proposed features as well as examine certain relevant features from the literature.

- ***Chapter 3: Extraction of Audio Features:***

  This chapter lists some of the relevant features that exist in the literature of speech/music discrimination and describes how they are computed. The chosen time domain features are related to the novel features presented in chapters 4, 5, and 6. Some frequency domain features are also presented for completion of the literature review and to illustrate their level of complexity compared to the simpler time domain features.

- ***Chapter 4: Ratio of Silent Frames:***

  This chapter introduces a novel feature called the Ratio of Silent Frames (RSF) and provides a thorough investigation of its performance compared to two other relevant features from the literature.

- ***Chapter 5: Time Series Events (TSE):***

  This chapter investigates the various possible events that could occur in any audio signal and shows a couple of important events that no one has investigated yet prior to the write up of this thesis. The chapter provides a thorough analysis of the two features in order to show that they can reach close to 100% speech/music discrimination.

- ***Chapter 6: Further Analysis of the Time Series Events:***

  Compared to the feature introduced in chapter 4, the features introduced in chapter 5 provided higher performance. Thus, this chapter provides further analysis of the TSE features that were discussed in chapter 5. Certain significant parameters of the TSE features are analysed in order to evaluate the significance of varying their values in the context of the speech/music discrimination problem.

- ***Chapter 7: Conclusion:***

  This chapter summarizes the outcomes and contributions of this thesis and provides some of the possible future work that could emerge from the contributed features.

1.7. Summary

The first chapter of this thesis provided an introduction to the thesis and how it fits in the wider relevant fields of research. It defined the main goal of this study and some of the motivations and applications where the outcome of this study could be applied. The methodology followed in producing this thesis was briefly described. The structure of the thesis was also explained describing the contents of each chapter and showing how it links to the rest of the chapters in the context of the speech/music discrimination problem.

The following chapter will provide details about the data used in this study to assess certain audio features from the literature as well as the proposed novel features.

Chapter 2: Research Requirements

2.1. Research Requirements

In order to carry out the experiments related to this research both speech and music audio data had to be either recorded or obtained from an authentic data supplier. Three obstacles have ledto the procurement of the audio data from professional suppliers instead of recording them in-house:

1. Lack of facilities at the School of Engineering at Durham University
2. In order to examine different types of music (genre) it was required to have professional musicians that can compose various types of music; e.g. classic, pop, jazz, etc.
3. In order to collect a wide range of speakers, it was required to have participating speakers from various age ranges and from both genders; something that would take time beyond the time frame of this project.

For the reasons above, it was more practical to purchase standard research audio data that meet certain criteria.

Two suppliers were identified that produced such data for the sole purpose of research. Each chosen data supplier recorded their audio data in an anechoic audio chamber, so that similar quality of sound was maintained when carrying out the comparison between speech and music. Also, equal sampling rate was ensured in order for this factor not affect the comparison. Nevertheless, the effect of changing the sampling rate was investigated in

chapter 6. More about the specifications of each type of data will be detailed in the following sections of this chapter.

2.2. Datasets

The core objective of this research is to extract and analyse certain audio features in the time domain. In order to carry out adequately rigorous analysis and to assess each audio feature proposed in this thesis, as well as some features from the literature for comparison, two types of datasets were acquired:

1. University College London (UCL) [40] speech dataset

2. Real World Computing (RWC) [41-43] music dataset

Both datasets are sampled at the same rate (44.1 kHz) in order to compare speech to music using any proposed feature. Furthermore, both datasets were recorded in an anechoic audio chamber. Such chambers are designed to be echo-free. Technically, the echo level cannot be zero.

The term Reverberation Time (RT) is used to define how low the echo is [44]. RT is an acoustical property of all architectural spaces. By convention, it is defined as the time taken for sound reflections to decrease by 60dB after the sound source ceases. Thus, it is sometimes referred to as RT_{60}. It can be computed from equation 2.1:

$$RT_{60} = k\frac{V}{\alpha S} \tag{2.1}$$

Where the parameters in the equation are defined as follows:

k: a constant; $0.163 sm^{-1}$,

αS: total effective surface absorption of a room; a sum of all the surface areas in the room multiplied by their respective absorption coefficients(α) which express the absorption factor of materials at given frequencies,

V: volume of the room

2.2.1. UCL Anechoic Chamber

The UCL anechoic chamber is shown in figure 2.1. Reflection of sound off the walls is reduced by inward-pointing pyramid shapes. The chamber is like a room inside another room; i.e. inner walls and outer walls. The UCL chamber used glass-fibre wedges. The outer walls are 33cm thick and the inner room is formed of metallic acoustic panels mounted on a floating floor. The inner wall of the chamber was separated from the outer wall by a cavity that is about 15cm wide. The room is also equipped with a ceiling mounted monitor used to prompt the speaker sitting behind the sound level meter.

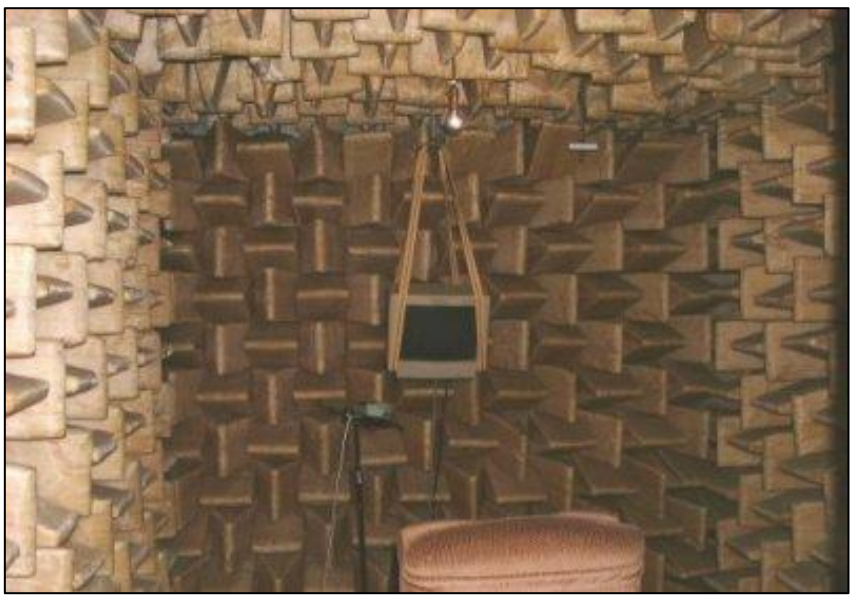


Figure 2.1 Speech recording at UCL anechoic chamber, [40]

The walls and ceilings are lined with 20cm×20cm×60cm wedges and the floor with 20cm×20cm×48cm wedges. Above the floor wedges are a gridded working area constructed from 60cm×45cm removable panels. These had a large 5cm grid size to minimise reflections. The reverberation time for the room is 20 milliseconds.

2.2.2. UCL Measuring Equipment

A Brüel & Kjær 2231 sound level meter was used for recording at a sample rate of 44.1 kHz. Its picture is shown in figure 2.2. The meter was placed approximately 30 cm from the mouth of the speaker and at 15° deflection on the transverse plane. Each speaker was asked to either read a certain text (e.g. a story, sentences, words) or to describe a scene after viewing it (e.g. cartoon). Recorded data were saved into audio files of the format ".wav".

A condenser microphone of type 4165 was attached to the sound level meter. The frequency response of the microphone is shown in figure 2.3. It can be observed that the microphone has almost a flat response. This is the case when positioned at 0° elevation. If the microphone is positioned at different elevations, then its response can be corrected according to the curves in figure 2.4. The microphone's specifications can be summarized as in table 2.1.The microphone measures the sound pressure level (SPL) in dB and any recorded voltage corresponds to a certain SPL. For the type 4165 condenser microphone, the voltage and SPL are related by equation (2.1) below:

$$\text{Voltage Precision} = 45.7 \text{ mV/Pa} \tag{2.1}$$

This means, 1 Volt corresponds to 21.88 Pascal; i.e. 21.88 N/m^2. However, the recorded reading is voltage, not pressure. It is always normalized to maximum 1 Volt and minimum –1 Volt.

(a) (b)

Figure 2.2 (a) Brüel & Kjær 2231 sound level meter; (b) type 4165 condenser microphone

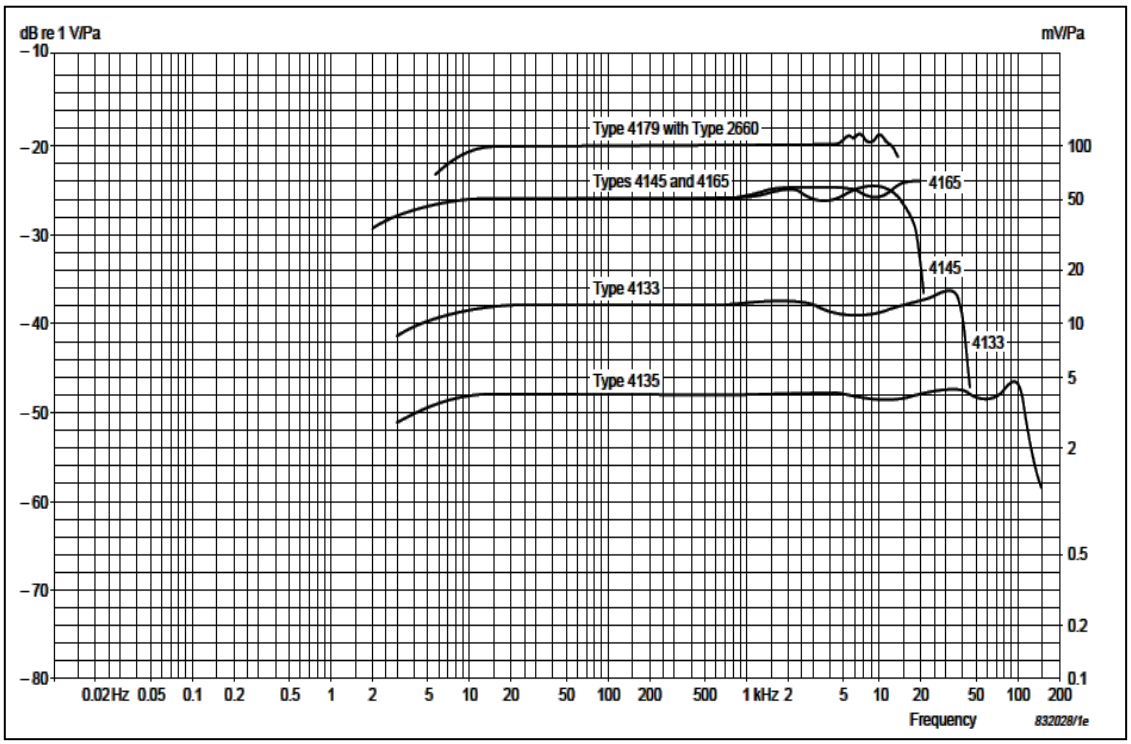


Figure 2.3 Frequency response for various microphones showing type 4165 condenser microphone

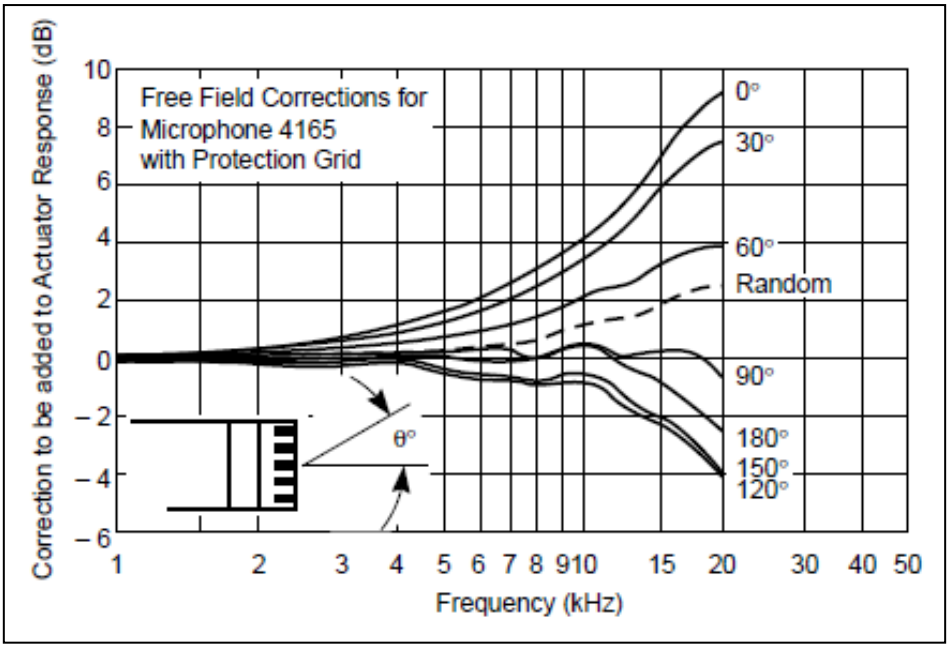


Figure 2.4 Correction to be added to the frequency response of the condenser microphone depending on its orientation

Sensitivity	45.7 mV/Pa
Frequency	2 Hz –20 kHz
Dynamic Range	24 – 146 dB
Temperature	– 10 to +50°C
Polarization	200 V

Table 2.1 Specifications of the condenser microphone used by UCL to record speech data

2.2.3. UCL Speech Dataset

After describing the chamber where the UCL data were recorded and the instruments used to record the data, this section describes the data acquired from UCL. The diversity of the data content in terms of gender and age was an important element which enriched the data making

the analysis suitable for all types of speech; i.e. any of the subclasses of speech are labelled as speech in general regardless of gender or age.

The database contains recordings of 45 speakers with South-Eastern English accent. The speakers were taken from both genders and involved both adults as well as children. The data involved words spoken by 18 women, 15 men, 6 girls, and 6 boys. The total length of the spoken individual words was about 95 minutes. That is equal to 5700 seconds of spoken words. All data were sampled at 44.1 kHz which covers the spectrum up to 22.05 kHz.

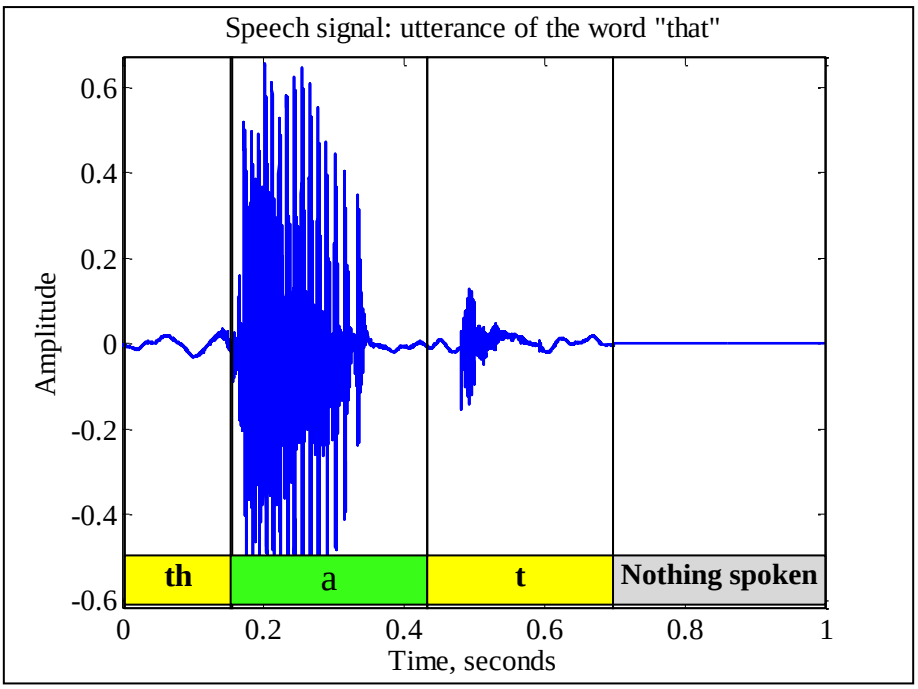


Figure 2.5 An audio signal from UCL dataset showing an utterance of the word “that”

Figure 2.5 shows an example where the word “that” was uttered by a male speaker. The various regions of the uttered word are clearly marked. It can be observed that the voiced segment has relatively higher energy than the unvoiced segments. For that reason, using energy alone to detect silence can lead to high misclassification error. The last segment in

figure 2.5 carries no information about the signal; this is where the speaker has finished speaking and null signal is recorded. Normally, such segments are discarded prior to the analysis. This can be done by segmenting the audio track into long-frames, e.g. 500 milliseconds. Such a frame is examined for silence. If it was found to be a silent frame, then it is discarded from the original sound track. Silence detection is discussed in more details in section 3.2.5 and in chapter 4.

2.2.4. RWC Music Dataset

The Music dataset was provided by Real World Computing (RWC) [41-43]. The data was recorded in an environment similar to that used by UCL; i.e. similar noise and echo levels. The dataset is composed of 315 musical pieces divided into 5 sets as structured below; sampled at 44.1 kHz.

1. **Popular Music Database (100 songs) &Royalty-Free Music Database (15 songs):**
    1. 80 songs in the style of Japanese popular music
    2. 20 songs in the style of Western popular music
    3. 5 well-known children's Japanese songs
    4. 10 well-known standard popular English songs
2. **Classical Music Database (50 pieces):**
    1. Symphonies: 4 pieces
    2. Concerti: 2 pieces
    3. Orchestral music: 4 pieces
    4. Chamber music: 10 pieces
    5. Solo performances: 24 pieces
    6. Vocal performances: 6 pieces

3. **Jazz Music Database (50 pieces):**
    1. Instrumentation variations: 35 pieces (5 pieces × 7 instrumentations)
    2. Style variations: 9 pieces
    3. Fusion (crossover): 6 pieces
4. **Music Genre Database (100 pieces)**
    1. 10 main genre categories (popular, rock, dance, jazz, Latin, classical, marches, world, vocals, and traditional Japanese music) and 33 subcategories: 99 pieces (33 subcategories × 3 pieces)
    2. A cappella (1 piece)
5. **Musical Instrument Sound Database (50 instruments)**
    1. A total of about 150 musical-instrument performances (generally 3 variations each for 50 types of musical instruments)

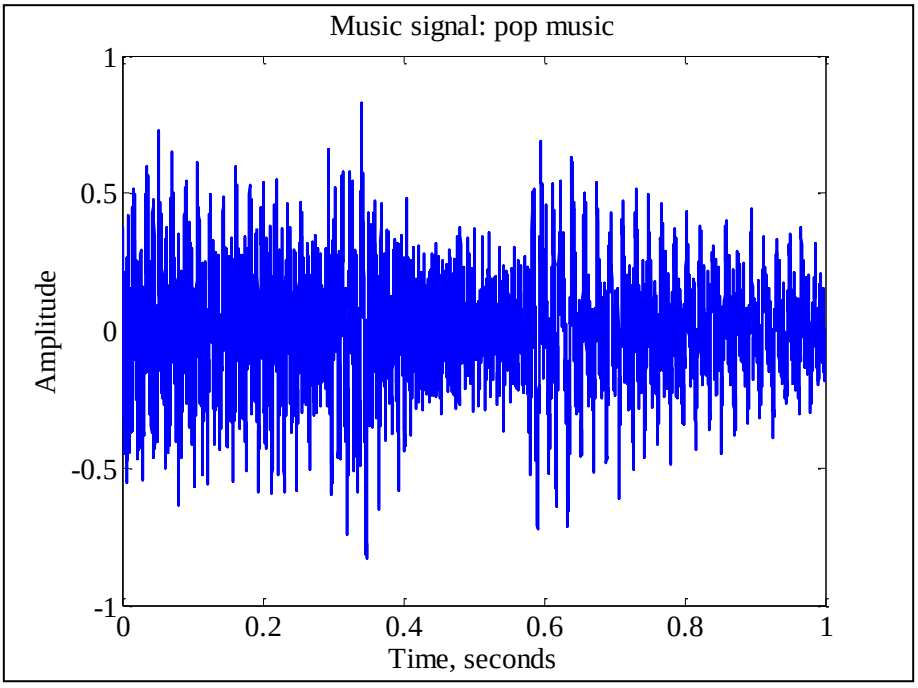


Figure 2.6 An audio signal from RWC dataset showing one second of pop music

The amplitude of both of the speech and music signals was normalized so that they range from minimum –1 to maximum +1. The pop music signal in figure 2.6 shows almost no silence at all and no segments with low energy. However, this may not always be the case. More rigorous analysis will be carried out later on in chapter 4 where the ratio of low energy frames will be investigated using what is called Low Energy Frames (LEF) feature.

The power spectral density of the speech and music signals is illustrated in figure 2.7 which is produced from the UCL and RWC data. In general, music has got higher power spectral density than speech.

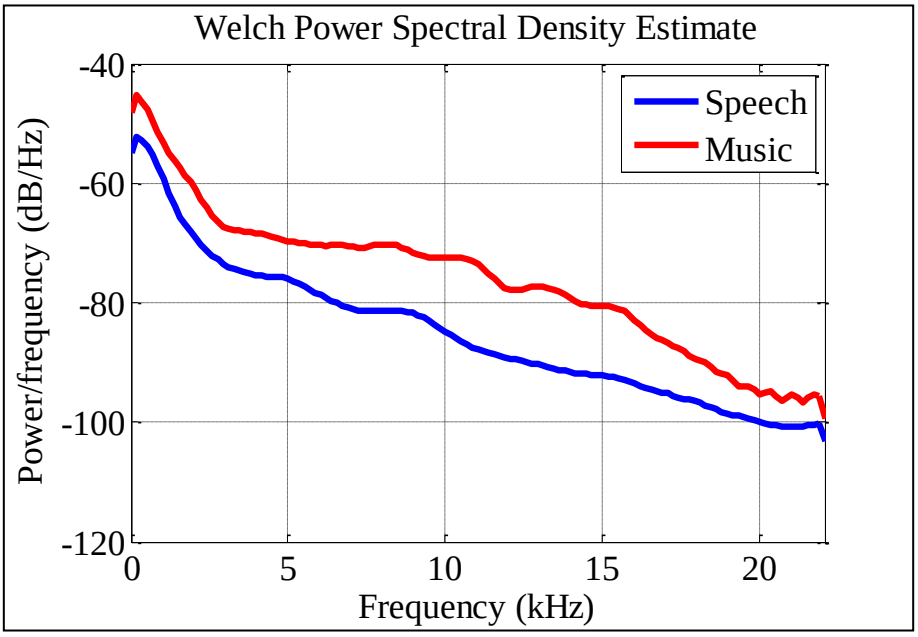


Figure 2.7 The power spectral density of speech and music

Most of the power is at the lower frequencies and it decays at the higher frequencies. Since the sampling rate used is 44.1 kHz, the maximum frequency that can be obtained is at half the sampling frequency, i.e. 22.05 kHz. The following can be observed:

1. The highest psd can be observed at about 500 Hz which is the fundamental frequency.
2. Throughout the spectrum, music always has higher psd than speech; this might vary from one signal to another, the statistical analysis of the variation of the psd at every frequency is not the subject of this study; a sample of speech and music is only shown here for demonstration.

2.3. Simulation Tool

MATLAB was used throughout this work to simulate and experiment with the datasets using the proposed methods. It was chosen due to its simple and powerful script programming. Although it is a generic programming tool, it is supported by several toolboxes associated with different fields of computations.

The Signal Processing Toolbox was the one applied for most of the experiments. It is very handy in importing the audio formats provided by the supplier, which was the 'wav' format. Each 'wav' file was read by the software and stored as a vector of samples for further processing. The toolbox has other capabilities like playing an audio file, filtering, and re-sampling it (either up-sampling or down-sampling). The built-in functions of MATLAB are not sufficient to carry out all of the tasks required for either the discrimination or the analysis of the proposed system. Certain other functions had to be scripted to simulate the prescribed features and to conduct the analysis afterwards. These scripts can be found in appendix A at the end of this thesis.

Listing 2.1 shows an example of a script that reads a "speech.wav" file, stores its raw time series samples as a vector "A" and its sampling frequency as a constant "f_s". If the signal is longer than 3 seconds, then only the first 3 seconds of the signal are saved into vector B,

otherwise the entire signal is saved into vector B. The script then plots the signal in the time domain.

Listing 2.1An example of a MATLAB script which reads an audio file and plots its waveform

```
% Read an audio file and save its raw data (A) and its
% sampling frequency (Fs)
[A,Fs]=wavread("speech.wav");
% Store up to 3 seconds of the signal into vector (B)
If length(A)> 3*Fs
     B = A (1: 3 * Fs);
Else
     B = A;
End
Time = (1 : length(B))/Fs;
plot(Time, B),
xlabel('Time, Seconds')
ylabel('Amplitude, Volts')
```

2.4. Summary

Chapter two has described in detail the data that were acquired and applied throughout the thesis for the assessment of all proposed audio features as well as any compared features from the literature. The chapter also provided a description of the environment where the data were recorded and its noise and echo levels. The instruments used to record the data were illustrated. Details of the types of music and instruments that composed the data were also provided.The following chapter will provide a background about the speech/music discrimination problem and how it was tackled in the literature.

Chapter 3: Audio Features

3.1. Introduction

This chapter is written to provide the reader with a background about what has been achieved so far in this field of research. The chapter mainly covers audio features that have been suggested by other researchers in order to be used for discriminating between speech and music.

In order to discriminate between speech and music signals, their raw samples are inadequate. They do not give discriminative information about their nature; i.e. being speech or music. Nevertheless certain features can be extracted from these raw data. The primary goal of every research in this field is to find such features. In this chapter, an overview of most of the features found in the literature will be provided for the reader to get an insight of the subject.

In most of the proposed discrimination systems, it is assumed that the signal is either speech or music. However, audio signals could also be classified into many other classes [21]; e.g. noise, silence, and mixture of speech and music. In such more complex situations one might consider the system to be a speech detector or a music detector. In that case, the system will mark the signal, with some certainty, as either speech or music.

3.2. Time Domain Features

The various audio features can be categorized into two broad domains as illustrated in figure 3.1: Time domain and Frequency domain. This section describes some of the temporal (time domain) features. For convention, it is important to define the structure of the audio signals analysed throughout this thesis.

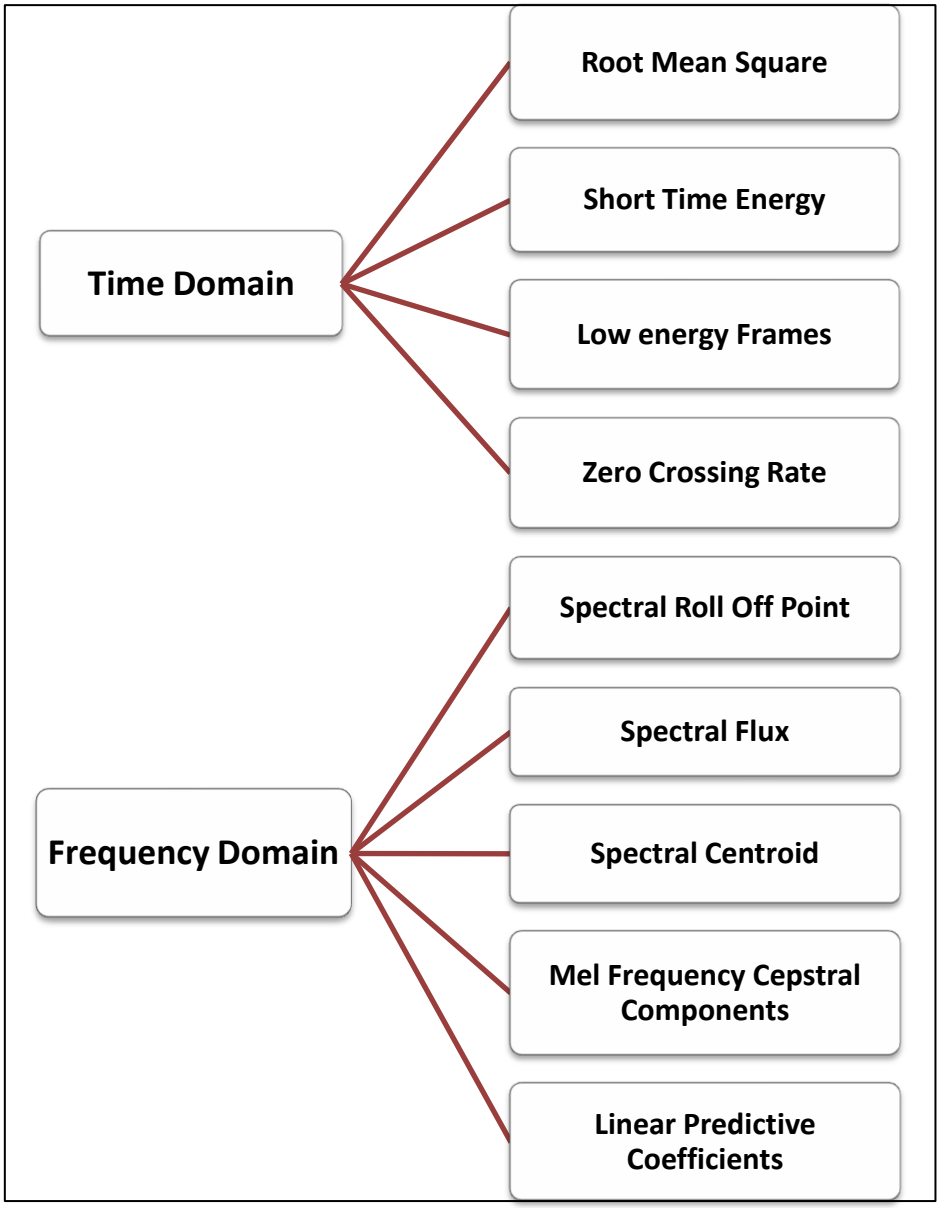


Figure 3.1 Audio features with examples from each category

Audio features are extracted at two levels [39]:

1. Short-term (frame) level which last about 10 to 30 milliseconds within which the audio signal is more stationary.
2. Long-term (track) level which go from 1 second up to tens of seconds. Such long intervals are also referred to in the literature as a “clip” or “window”. Throughout this thesis, the term track is adopted.

Both levels are illustrated in figure 3.2. A long track that has a stream of N samples is split into a number of frames M. Each frame would then contain R samples. While some applications consider a certain amount of overlap between consequent frames, other applications do not.

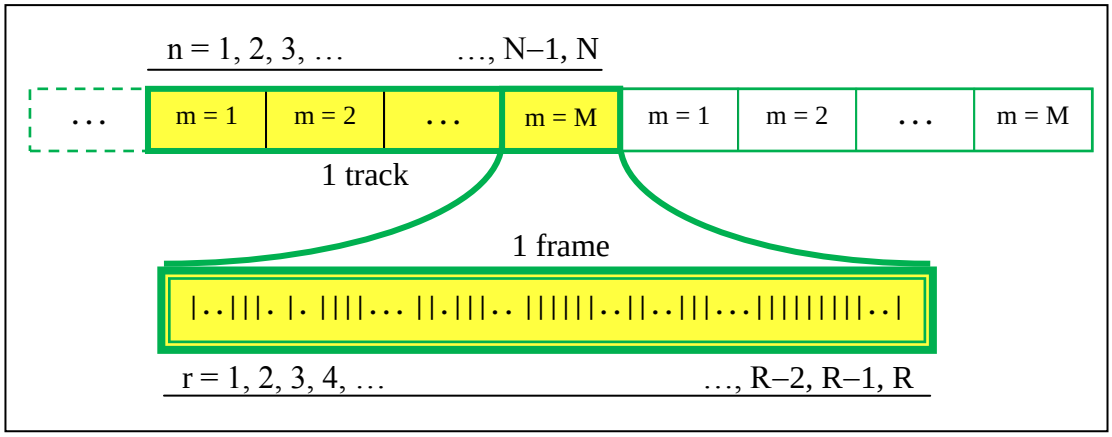


Figure 3.2 A stream of tracks that is split into M frames; each frame has R samples

For example, when sampling at f_s = 44.1 kHz, a track of 5 seconds would contain N = 5 × 44100 = 220500 samples. If each track is split into 20 millisecond frames, then each frame would contain R = 20 × 10^{-3}× 44100 = 882 samples and there will be M = 250 frames in each 5 seconds track.

3.2.1. Root Mean Square (RMS)

The root mean square (RMS) of a signal is an indication of the power content in the signal. The RMS power of a particular audio frame can be found using [31] equation 3.1:

$$\text{RMS} = \sqrt{\frac{\sum_{n=1}^{N} x^2(n)}{N}} \tag{3.1}$$

Where N is the number of samples in each frame and x(n) is the audio signal at sample n.

The RMS is a feature that is used at the frame level instead of the track level. RMS is used in the literature to identify other features as will be evident in chapter 4. The RMS power is sometimes loosely called by different names; e.g. volume, loudness, and energy. The latter name (i.e. energy), however, is normally computed as described in the following section.

3.2.2. Energy

The energy of any signal $x(n)$ is a very fundamental characteristic [45, 46]. It can be computed using equation 3.2.

$$E \equiv \sum_n |x(n)|^2 \tag{3.2}$$

Regardless of the scale on the energy axis, it can be observed that most of speech frames have relatively low energy. Music, on the other hand, is not restricted in the low level range; it exists in almost every range of energy.

3.2.2. Percentage of Low Energy Frames (LEF)

This is a feature that can be extracted in the time domain. Splitting a sound track (e.g. 1 second long) into small frames (e.g. 20 ms) would produce a number of frames (e.g. 50

frames). The RMS power of each frame is then computed using equation 3.1. The mean RMS is then computed. Low energy frames are then labelled. A low energy frame is a frame whose RMS is less than 50% of the mean RMS; i.e. 0.5 RMS_{ave}.

This feature was exploited by Scheirer and Slaney [30] and by Saad *et. al.* [21]. The latter showed performance of correct classification for speech to be 95% and for music to be 85%. General performance of speech and music discrimination was shown to be 90% indicating 10% general misclassification.

If a threshold is used to discriminate between speech and music (e.g. at LEF = 40%) then up to 96% of speech frames can be correctly labelled as speech while about 30% of music will be misclassified as speech. There is always a trade-off between correct classification and misclassification. Section 3.5 will provide more discussion about how to measure this trade-off and how to choose the optimum threshold for best performance of any feature.

The answer to "how is the optimum threshold chosen for the LEF feature?" is presented in chapter 4 where the LEF feature is compared to other features.

3.2.3. Modified Low Energy Ratio (MLER)

The previously mentioned audio feature (LEF) was modified by Wang *et. al.*[23] to represent a single feature system that provided more than 97% classification accuracy in spite of its low computational load.

The modified feature was named "Modified Low Energy Ratio" (MLER); and was computed using the equation:

$$MLER=\frac{1}{2M}\sum_{m=1}^{M}\left[Sgn\left(lowthres - E(m)\right) + 1\right] \quad (3.3)$$

$$lowthres = \delta \frac{1}{M} \sum_{m=1}^{M} E(m) \quad (3.4)$$

where: M is the total number of frames in a track, $E(m)$ is the short time energy of the m^{th} frame, and δ is a control coefficient which decides how low $E(m)$ needs to be so that the frame is considered as "low energy".

$Sgn(x)$ is by convention defined as below:

$$Sgn(x) = \begin{cases} +1 & , x > 0 \\ 0 & , x = 0 \\ -1 & , x < 0 \end{cases} \quad (3.5)$$

While δ was set to 0.5 in the LEF feature, it is varied in the MLER and a range of [0.05 to 0.12] was recommended. While (δ = 0.5) provided 11.75% average misclassification, (δ = 0.1) provided about 4% average misclassification; as far as the test data used by Wang *et. al.* in [23].

3.2.4. Zero Crossing Rate (ZCR)

This feature was first introduced by Kedem in [16] who related it to the dominant frequency of a signal. It is a count of the number of times an audio signal crosses the level of zeroamplitude in the time domain. Prior to counting, the mean of the signal is deducted in order to remove any DC component from the signal.

Assuming a discrete signal with a series of samples [S_1, …, S_N] with zero mean, the following set of equations were used by Kedem to count the number of times the signal crosses the level of zero amplitude:

Putting the mean (zero) as the threshold that the signal crosses,

$$X_t = \begin{cases} 1, & S_t \geq 0 \\ 0, & S_t < 0 \end{cases}, \; t = 1, 2, \ldots, N \tag{3.6}$$

The function:

$$d_t = (X_t - X_{t-1})^2 \tag{3.7}$$

is an indicator of whether a crossing occurred ($d_t = 1$) or not ($d_t = 0$). Thus, the ZCR is simply:

$$D = \sum_{k=2}^{N} d_k \tag{3.8}$$

For a single sinusoidal frequency, there are two crossings; thus 2 Crossings/Cycle. Assume that within a cycle of T seconds, there are D crossings. The duration of that cycle can also be expressed as $(N-1)\times\Delta t$, where Δt is the sampling period. Hence, the following relation:

$$\text{Number of crossings per second} = \frac{D}{(N-1)\Delta t} \text{ crossings/second} \tag{3.9}$$

Frequency is defined as the number of cycles/second. Therefore, one can use units to conclude the frequency as follows:

$$\text{Frequency} = \frac{D}{(N-1)\Delta t} \text{ crossings/second} \div 2 \text{ crossings/cycle} \tag{3.10}$$

$$f = \frac{D}{2(N-1)\Delta t} \text{ cycles /second} = \frac{D}{2(N-1)\Delta t} \text{ Hz} \tag{3.11}$$

$$\text{Radial Frequency} = 2\pi f = 2.\pi.\frac{D}{2(N-1)\Delta t} = \frac{\pi D}{(N-1)\Delta t} \text{ rad/second} \tag{3.12}$$

The dominant digital frequency is then computed using:

$$\theta = \pi D/(N-1) \text{ rad} \tag{3.13}$$

The sampling period in practice is computed from the sampling frequency, f_s, using:

$$\Delta t = 1/f_s \tag{3.14}$$

Therefore, if the number of zero crossings is to be used for estimating the dominant frequency, one must consider the sampling frequency:

$$f = \frac{D}{2(N-1)} f_s \text{ cycles sec}^{-1} \text{ (Hz)} \tag{3.15}$$

To illustrate the usage of ZCR for the estimation of normalized[1] dominant radial frequency, we used an example of a signal with two distinct frequencies:

$$Z_t = A \cos(\omega_1 t) + B \cos(\omega_2 t) \tag{3.16}$$

Given the radial frequencies:

$$\omega_1 = 1 \text{ rad/sec}; \qquad \omega_2 = 3 \text{ rad/sec}$$

Consider the following three cases of A and B:

i. Neither frequencies is dominant (A = B = 10) results in a dominant frequency ω_0 = 2.147 rad/sec which lies somewhere between the two frequencies; figure 3.3.
ii. One frequency is more dominant than the other (A = 2; B = 10) results in a dominant frequency ω_0 = 2.9679 which is much closer to the dominant frequency (figure 3.4).
iii. A low pass filter is used to remove the higher frequency (A ≈ 2; B ≈ 0) resulting in 0.9946) as expected.

[1] Normalization implies assuming unit sampling period, where all frequencies are divided by f_s; some text refer to it as digital frequency.

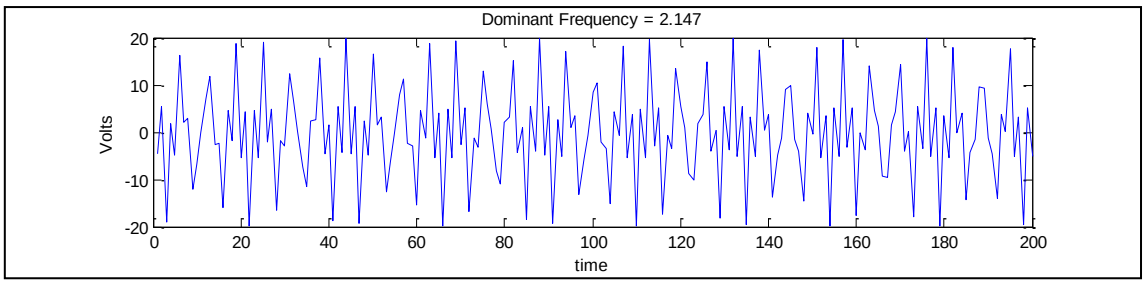


Figure 3.3 Case (i) where both frequency components are equally dominant

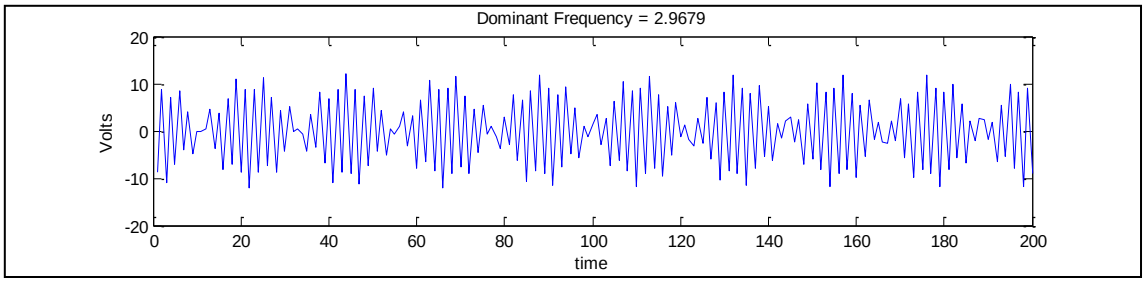


Figure 3.4 Case (ii) where one frequency is dominant

The ZCR is a fast and approximate method to estimate the dominant frequency while avoiding the spectral analysis like the Fourier Transform. However, it has two significant drawbacks:

1. it gives no information about the rest of the spectrum (the less dominant frequencies.)
2. if two frequencies are equally dominant, the ZCR gives a third frequency that lies somewhere in between (average), while giving absolutely no information about the real spectral content.

The ZCR is not useful when used alone. However, it could be exploited in association with other features. For example, the correlation between the ZCR and the RMS of audio frames was exploited in [31]. While the ZCR and RMS are independent for music, they are somehow

correlated for speech. A normalized correlation function was defined to compute the correlation between the two features as in the equation below.

$$C_z = \frac{\sum_{i=1}^{N} A(i)Z(i)}{2A_x - A_n - A_m} \tag{3.17}$$

Where $A_x$is the maximum RMS among all frames within an audio track, A_n is the minimum RMS among all frames within an audio track, and A_m is the median RMS of the frames constituting an audio track. If C_z is close to zero, then the track is classified as speech, otherwise it is classified as music. An empirical threshold is used for this classification. Using such correlation algorithm 90% of music was correctly classified while 60% of speech was correctly classified.

3.2.4. Pitch

The fundamental frequency for any audio signal is called pitch. This feature cannot be accurately measured, but can be estimated. Different methods exist that can be used to estimate the pitch of an audio signal. Each method has a different accuracy and complexity, therefore the choice is application dependent. Musical instrument sounds can be identified from their pitches [27].

In general the pitch information of an audio signal can be extracted using either temporal or spectral analysis. While the pitch frequency for speech is in the range 50-450 Hz, it is much wider for music [39].The maximum anticipated frequency range of pitch is that for female speech (500-600 Hz). Therefore, the speech is normally low-pass filtered at a frequency of 1 kHz. Thus, it is normally down sampled to 1 kHz [45].

A number of methods exist that can be used to estimate the pitch of an audio signal. The details of each method can be found in the corresponding references provided. Some of the methods are:

1. Gold-Rabiner pitch extractor [47]
2. Autocorrelation methods [39]: The Autocorrelation Function and the Average Magnitude Difference Function (AMDF)
3. Simplified Inverse Filter Tracking (SIFT) Algorithm [48]

Much more can be found about pitch determination of speech signals in particular from [49]. Examination of the pitch contour of any track after computing the pitch of each frame could lead to various speech/music discrimination methods. For instance, non-pitch-ratio (NPR) is the "percentage of frames without pitch" [39]; a feature that can measure the ratio of unvoiced or noise frames. Apart from the NPR, Liu et. al. [50] utilized two other features to capture the pitch variation:

1. Similar Pitch Ratio (SPR): the percentage of frames in an audio track that has a pitch value that is similar to the previous frame. When neighbouring frames have similar pitch, a smooth pitch contour is obtained for voiced or music frames within a track.
2. Pitch Standard Deviation (PSTD): this feature measure the level of variation within the pitch contour.

3.2.5. Silence

When listening to any news radio channel it can be observed that in every talk there is a lot of pauses. Silence is not a feature, but rather a general term that is used to imply, theoretically,

zero energy in a particular frame. In practice, a threshold is defined below which a frame is considered to be silent. However, energy alone is not sufficient to define silent frames. Speech can be classified into voiced speech (e.g. the segment "ee" in the word "sheet") or unvoiced speech (e.g. the segment "sh" in the same word) [45]. Because unvoiced speech has low energy their frames would be classified as silent frames.

In terms of tracks and frames, one can realize that every track would contain a number of frames that are silent. Music, on the contrary, has far less silence than speech. Regardless of the method used to identify silent frames within a track, this feature is always beneficial and could be exploited to discriminate between speech and music.

Liu et. al. [50] proposed a feature called non-silence-ratio (NSR) where they identified the frames within a track that were not silent and computed the ratio of these frames to the total number of frames within a track. Low energy unvoiced speech frames could be classified as silent frames. To avoid that, they incorporated the ZCR feature beside the volume to identify silent frames. They used two preset thresholds: one for what they called volume (RMS) and another one for the ZCR. A frame was considered to be silent if the volume (RMS) was below its preset threshold and the ZCR was below its preset threshold. If both conditions were satisfied, then the examined frame was considered to be silent. By computing the ratio of non-silent frames, they obtained what they called the non-silence-ratio (NSR) feature to discriminate between speech and music. A silence-related feature is presented in chapter 4 which provides further enhancement and has some advantages over the NSR feature.

Silence was also exploited by Panagiotakis and Tziritas [31] by detecting silent intervals using the RMS and the ZCR features such that any frame that fulfils any of the following criteria is classified as a silent frame:

1. RMS is less than threshold T_1
2. RMS is less than 10% of the maximum RMS or less than the threshold T_2
3. ZCR = 0

After detecting the silent intervals, the neighbouring silent intervals are grouped. Audible (non-silent) intervals are also grouped. Silent Intervals Frequency (F_v) is defined as the number of silent intervals in each segment (track). Setting an empirical threshold it was reported by Panagiotakis and Tziritas [31] that almost always for speech $F_v > 0.6$, while for 65% of music tracks, $F_v < 0.6$.

3.3. Frequency Domain Features

Some researchers have chosen to exploit the spectral features of audio signals [51]. Some of these features are explained in the following subsections. Features from the frequency domain are not the subject of this thesis, but are briefly mentioned here for completion of the literature review as well as to show the relative level of computational load in comparison to the time domain features.

Prior to computing any of the frequency domain features, the signal can be transformed from the time domain to the frequency domain using the Discrete Fourier Transform (DFT). The Inverse Discrete Fourier Transform (IDFT) can be used to transform the signal back to the time domain, but is normally not necessary in the context of feature extraction. Both the DFT and IDFT can be obtained as in equations (3.18) and (3.19), respectively [46]:

DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N} \qquad \text{k = 0, 1, 2, . . . , N–1} \qquad (3.18)$$

IDFT:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j\,2\pi kn\,/N} \qquad \text{n = 0, 1, 2, . . . , N–1} \qquad (3.19)$$

The relative level of complexity is evident from this transform when compared to the time domain features which are directly taken from the time series of samples. Further computational load is added to each frequency domain feature by the feature extraction algorithm detailed in the following subsections.

3.3.1. Spectral Roll-off point (SR)

This feature was introduced by [30] and is defined in [21] as a means of measuring the frequency below which 95% of the power resides. Mathematically, this is expressed using:

$$\text{SR = K,} \qquad \text{where } \sum_{f<K} X[f] = 0.95 \sum_{f} X[f] \qquad (3.20)$$

Where $X[f]$ is the power of the audio signal at the corresponding frequency f, and K is the spectral roll-off point on the frequency axis that satisfy the condition in equation 3.20. Using this feature, Saad et. al. [21] obtained a low performance of 65% discrimination between speech and music.

3.3.2. Spectral Flux (SF)

The summation of the differences between adjacent samples in a signal's spectrum for a single frame is known as the spectral flux [21]. Once computed using:

$$SF = \sum_{k} \| |X[k]| - |X[k-1]| \| \qquad (3.21)$$

The average of the spectral flux for all frames is then computed as the feature to be extracted.

Exploiting such feature resulted in 89% correct speech/music discrimination when investigated by Saad et. al. [21] to classify their data.

3.3.3. Spectral Centroid (SC)

As the name indicates, this feature represents the central point in the signal's spectral power distribution in a frame of samples. On average, the SC for speech is low compared to that for music. Also, the SC for voiced speech is lower than for unvoiced speech signals.

SC for a frame of an audio signal can be computed as:

$$SC = \frac{\sum_k kX[k]}{\sum_k X[k]} \tag{3.22}$$

Where k is an index corresponding to a frequency (or a band of frequencies) whose power is $X[k]$. Furthermore, better results were obtained by [21] when using the second moment; i.e. replacing k by k^2 as in the following equation:

$$SC = \frac{\sum_k k^2 X[k]}{\sum_k X[k]} \tag{3.23}$$

An average performance of 82.5% was obtained when using the second moment to compute the spectral centroid.

The spectral centroid is sometimes also called "brightness". Although most researchers attempt to investigate and thoroughly analyse individual features, they tend to combine/fuse features into one algorithm [21, 30, 52]. Sound-fisher [53], for example, is a sound-retrieval-software that is commercially available for customers. It makes use of certain audio features

(e.g. pitch, loudness, and brightness) in order to “fish” within a given database for audio files that “sound like” a given audio file.

3.4. Assessment and Comparison

The lack of a unified systematic evaluation procedure for all discrimination methods makes it practically infeasible to compare newly proposed features to every other feature in the literature. Furthermore, each researcher uses a different dataset to assess his work. Thus, for an ideal and comprehensive comparison, every newly proposed feature should be tested against older ones using the same dataset. Such exhaustive examination of every feature is beyond the time frame of this thesis. Alternatively selected relevant features are examined.

The solution of any discrimination problem has three dimensions: feature extraction, definition of an algorithm, and choosing a classifier. An idealcomparative study, which is not the objective of this thesis,should compare all previously proposed features and algorithms against each other using the same datasets while maintaining the same criteria (e.g. track length, frame length, number of test tracks, sampling rate, echo level, and noise level) in order to identify the superiority of one feature (or set of features) over the others. However, a feasible goal that this thesis attempts to achieve is to fill a gap of research that has been left without discussion. Moreover, certain relevant features were selected to carry out some comparison with the proposed features for the purpose of assessment.

The aim of this thesis is to present a set of novel features for the purpose of speech/music discrimination in the time domain and to investigate the performance of using each one of them individually.

Throughout this thesis, each relevant audio feature is examined using the same dataset to compare their individual performance to discriminate between speech and music digital audio signals. Further work could be carried out beyond this thesis to investigate the optimum set of features that suits a certain application. Real-time applications (e.g. speech recognition systems, hearing aids, etc.) would require features with low computational load and low latency. Data storage and retrieval applications, on the other hand, do not have such restrictions on time and thus would only be limited by implementation cost (e.g.software and hardware). The features proposed in this thesis were found to be more suitable for data retrieval.

3.5. ROC Curves

A Receiver Operating Characteristics (ROC) curve is a visual technique that is used to evaluate classifier performance [54, 55]. Assume a binary classifier that detects one of two types of signals (e.g. positive and negative). The problem is considered as a detection problem; i.e. detecting one type (e.g. positive) with the possibility that the other type is also detected by mistake.

Figure 3.5 shows an example of a certain feature whose probability of occurrence (i.e. distribution) is plotted against the feature's magnitude for speech and music signals. There is a clear overlap between the two types of audio signals. The overlap region contains certain probability of detecting either type by mistake while aiming to find the other one.

Thus, there is always a trade-off between true positive rates[1] (correctly detecting a positive signal) and false positive rates[2] (falsely detecting a negative signal). These are analogous to

[1]Also known as hit rate
[2]Also known as false alarm

the detection of either speech or music. Any speech detection should be assessed by its true positive rate (correctly detecting a speech signal) and false positive rate (falsely detecting a music signal); and vice versa for music detection.

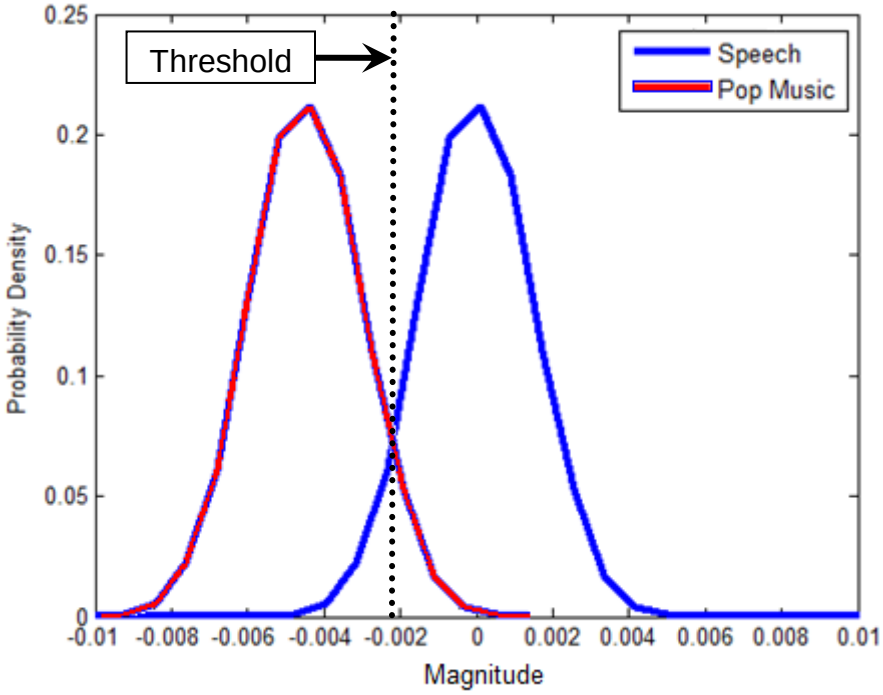


Figure 3.5 Example of the overlap between the distributions of two features

The true positive rate (TP) of a classifier is estimated using:

$$\mathrm{TP} \approx \frac{\text{Positives Correctly Classified}}{\text{Total Positives}} \tag{3.24}$$

The false positive rate (FP) of a classier is estimated using:

$$\mathrm{FP} \approx \frac{\text{Negatives Incorrectly Classified}}{\text{Total Negatives}} \tag{3.25}$$

Figure 3.5 shows a threshold that is used to discriminate between speech and music. Placing the threshold at each point along the x-axis and sweeping the whole range of magnitudes

would give a pair of *TP* and *FP* at each threshold. Plotting the *TP*versus the *FP* would produce the ROC curve for the examined feature. The ROC curve technique will be used for evaluating the performance of detecting either speech or music using various types of features throughout the rest of the thesis. Appendix A shows listings A1 and A2 used to compute the ROC curves for any pair of distributions.

Both TP and FP have values ranging from 0 to 1. An ideal classifier should have TP =1 and FP = 0. Practically, this scenario is rarely met. Nevertheless, if the TP is plotted against the FP, a graph/curve can be obtained that illustrates the relation between the two values as shown in figure 3.6.

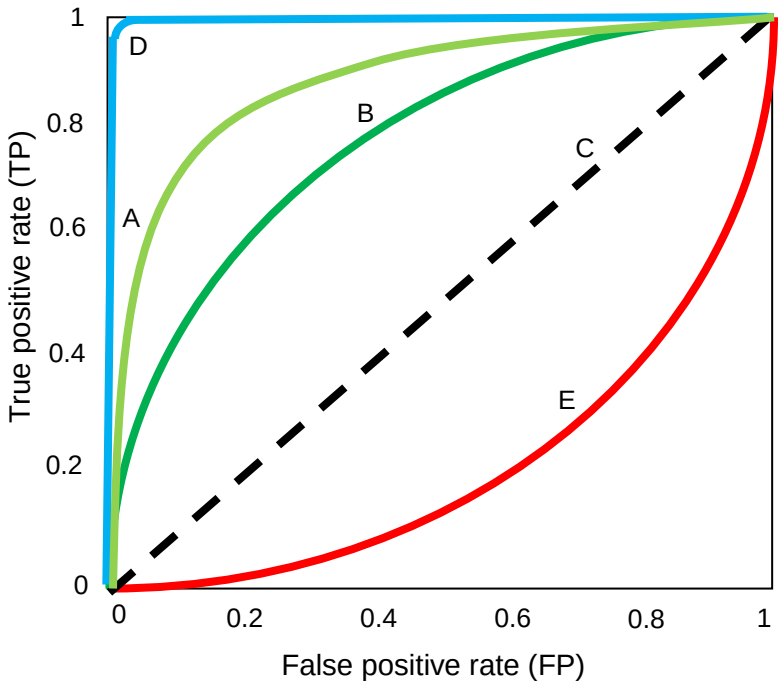


Figure 3.6 The space of ROC curves

The significance of certain points on the ROC curves is as described below:

1. **Point A** shows 60% hit rate with 5% false alarm
2. **Point B** has a higher hit rate (75%) but at the expense of higher false alarm (40%)
3. **Point C** has equal hit rate and false alarm (70%); this is referred to as random performance. Any classifier that falls on this line performs no better than a random toss of a coin.
4. **Point D** is an ideal situation where the signal is correctly classified 100% of the time.
5. **Point E** falls below the random performance. Any classifier that falls in this region of the space performs worse than a random guess and should not be used.

3.6. Summary

Chapter 3 provided a background about the audio features that exist in the literature both in the time domain and the frequency domain. It also discussed the methodology that will be used to assess any proposed feature in order to compare with other features from the literature.

The following chapter will present the first novel feature that is contributed to the field of speech/music discrimination and evaluate its performance.

Chapter 4: Ratio of Silent Frames

4.1. Energy of Audio Frames

Thorough analysis of the energy of audio frames has led to the exploitation of this fundamental feature. In this chapter two features from the literature are discussed: the percentage of Low Energy Frames (LEF) feature and the Modified Low Energy Ratio (MLER). In this chapter, a novel feature is proposed and compared to these two features: the Ratio of Silent Frames (RSF).

The proposed feature is based on the fact that there are more silent frames in a speech audio track than in a music audio track [50]. The energy and the zero crossing rate of a certain frame are used to detect silent frames. The ratio of silent frames (RSF) gives indication of whether the analysed signal is speech or music. ROC curves are then used to compare the performance of the LEF, MLER and RSF features.

4.2. Percentage of Low Energy Frames (LEF)

This is a feature that can be extracted from sampled sound tracks in the time domain. The energy of a signal x(r) that has R samples can usually be obtained from equation (4.1):

$$E \equiv \sum_{r=1}^{R} |x(r)|^2 \tag{4.1}$$

However, the word “energy” is loosely used in the literature and the energy of a signal is defined by its Root Mean Square (RMS) power instead. The latter being an indication of the

frame energy. The term "volume" is sometimes used instead and is computed from the RMS power [50].The codes scripted to compute the RMS power and the energy of any signal can be found in Appendix A; listings A3 and A4, respectively.

The LEF feature is defined as the proportion of frames with RMS power less than 50% of the mean RMS power within a one second window [21]. It is also used in [56] under the name "Low Short-Time Energy Ratio" (LSTER). The code scripted to compute the LEF feature of any input signal is shown in listing A5.

First, a sound track, x[n] consisting of N samples, where n is the sample number, is split into a series of smaller frames consisting of R samples each to give a fixed number of frames, M. This is illustrated in figure 3.2.

For example, splitting 1 second track, consisting of 44100 samples into M = 50 frames gives frames of 20 ms duration. At a sampling frequency f_s = 44.1 kHz, each frame will consist of $R = 20 \times 10^{-3} \times 44100 = 882$ samples.

The root mean square of the m^{th} frame, RMS_m, is then computed using equation (4.2):

$$RMS_m = \sqrt{\frac{\sum_{r=1}^{R} x^2(r)}{R}}, \qquad \text{where} R = \frac{N}{M} \tag{4.2}$$

The mean RMS of the total M frames (i.e.$\overline{RMS}$) is then computed using equation (4.3):

$$\overline{RMS} = \frac{1}{M} \sum_{m=1}^{M} RMS_m \tag{4.3}$$

Low energy frames are those frames that satisfy the criteria:

$$RMS_m < 0.5\ \overline{RMS} \tag{4.4}$$

This feature was exploited by Saad et. al. [21] and Scheirer and Slaney [30]. On their test data, Saad et. al. reported 5% error on speech detection and 15% error on music detection; i.e. 10% general classification error. On the other hand, Scheirer and Slaney reported 14 % ± 3.6 % error of classification; i.e. between 10.4% and 17.6%. The discrepancy between these two results could be due to not unifying the method of assessment and the tested data. Variation in sampling frequency, frame duration, overlap between frames, and noise level in each dataset are parameters that could influence their outcome.

4.3. Modified Low Energy Ratio (MLER)

Wang et. al [23] had an attempt to enhance the LEF feature. They investigated the effect of using a different threshold. Instead of using 0.5 $\overline{RMS}$ as a threshold they scanned the whole range of possible thresholds in order to find the optimum range replacing the coefficient 0.5 by a control coefficient called δ. They named the modified feature "Modified Low Energy Ratio" (MLER) and computed its value using the following set of equations.

$$MLER = \frac{1}{2M}\sum_{m=1}^{M}\left[Sgn\left(lowthresh - E(m)\right)+1\right] \tag{4.5}$$

$$lowthresh = \delta\frac{1}{M}\sum_{m=1}^{M}E(m) \tag{4.6}$$

Where: E(m) is the short time energy of the m^{th} frame, δis a control coefficient which decides how low E(m) needs to be so that the frame is considered as "low energy", and Sgn(x) is by convention defined as below:

$$Sgn(x) = \begin{cases} +1 & , x>0 \\ 0 & , x=0 \\ -1 & ,x<0 \end{cases} \tag{4.7}$$

Instead of using the RMS power, Wang et. al. used the short time energy in order to mark low energy frames. The code scripted to compute the energy is shown in listing A4.

While δwas fixed to 0.5 in the basic LEF feature, it was varied in the MLER and a range of [0.05 to 0.12] was recommended. The code scripted to compute the MLER feature of any signal is shown in listing A6. The data tested by Wang et. al [23] demonstrated an enhancement of about 8% in the average misclassification as compared to the basic LEF feature. Their contribution involved the change of the control coefficient (δ) and using the short time energy instead of the RMS power.

4.4. Ratio of Silent Frames (RSF) Feature

When an audio track is split into a number of frames (e.g. 50 frames) a certain number of frames are found to carry no sound (i.e. silent frames). The fact that there are more silent frames in speech than in music can be exploited to discriminate between speech and music. The Zero Crossing Rate (ZCR) of a particular signal corresponds to the number of times a signal crosses its zero level in one second [16]. Furthermore, it is known that the ZCR for silent frames is far less than any other type of audio frames [25].

Splitting any one-second audio track into M frames, each containing R samples, and has D_m zero crossings; its ZCR_m can be computed as given in equation (4.8):

$$ZCR_m = \frac{D_m}{R-1} \quad \text{crossings/sample} \tag{4.8}$$

After computing the ZCR of the m^{th} frame that has R samples, its energy E_m is computed using equation (4.9):

$$E_m \equiv \sum_{r=1}^{R} |x(r)|^2 \tag{4.9}$$

The energy could also be represented by the RMS power as in the LEF feature; i.e. using equation (4.2):

$$RMS_m = \sqrt{\frac{\sum_{r=1}^{R} x^2(r)}{R}} \quad , \qquad \text{where} R = \frac{N}{M} \tag{4.2}$$

Then the product of the two parameters, E_m and ZCR_m, is computed using equation (4.10):

$$S_m = E_m \times ZCR_m \tag{4.10}$$

The threshold which determines whether a particular frame has low energy or not is then computed using equation (4.11):

$$S_{th} = \delta \frac{1}{M} \sum_{n=1}^{M} S_n \tag{4.11}$$

where δ is a control coefficient whose value ranges from 0 to 1.

The whole range of δ was examined and found that the values ranging from 0.05 to 0.15 give the best performance of this feature; this matches with the MLER range of δ. The value 0.1 was chosen to compare with LEF and MLER. The last step is to compute the RSF parameter using equation (4.12).

$$RSF = \frac{1}{2M} \sum_{m=1}^{M} \left[Sgn\left(S_{th} - S_m\right) + 1 \right] \tag{4.12}$$

where Sgn(x) is computed from equation (4.7).

Low energy unvoiced speech frames could be misclassified as silent frames. To avoid that, Z. Liu *et. al.*[50] incorporated the ZCR feature beside the energy to identify low energy frames. They had two preset thresholds: one for what they called volume (RMS) and another one for the ZCR. A frame was considered to be silent if the volume (RMS) was below its preset

threshold and the ZCR was below its preset threshold. If both conditions were satisfied, then the examined frame was considered to be silent. By computing the ratio of non-silent frames, they obtained what they called the Non Silence Ratio (NSR) feature to discriminate between speech and music. At a first glance, the method used by Z. Liu *et. al* .[50] seems similar to the one proposed in this chapter. Thorough comparison between the two methods implies the following discrepancies:

1. The RSF algorithm uses the short time energy while the NSR uses the volume (RMS). While the latter is only an indication of energy, the former is an exact measure of the frame's energy.
2. The RSF algorithm uses one parameter (S) while the NSR uses two separate parameters (RMS and ZCR). This implies that NSR needs three steps: verifying the criteria (threshold) of the first parameter (i.e. RMS threshold), verifying the criteria (threshold) of the second parameter (ZCR threshold), and finally verifying that both criteria are met. While the first two criteria can be examined in parallel, the third criteria cannot be verified until both of the first two parameters are verified. The RSF feature, on the other hand, computes the two parameters (this can be done in parallel), computes the S parameter and then verifies the criteria (threshold) of the S parameter.
3. The most significant difference between the RSF and NSR is the generality of RSF feature to any corpus of audio data. The two thresholds suggested by Z. Liu *et. al*.[50] are fixed to preset values that suit their data and might not work with different datasets; i.e. they need to be re-examined every time a new dataset is processed. On the other hand, for the RSF feature the threshold is a ratio of the mean value of the examined data. Nevertheless, both LEF and MLER features did not consider the ZCR feature.

Every study uses a different data set to assess its hypotheses. Comparing a newly proposed method to the methods that exist in the literature is usually hindered by this fact. For instance, using different sampling rates could influence such comparison, in spite of resampling later on to match the sampling frequency between the two data sets. For that reason, we opted to use data sets that were acquired at the same sampling frequency and recorded in a similar environment in order to demonstrate the performance obtained by using each one of the three features: LEF, MLER, and RSF. Listings A8, A9, and A10 show the codes scripted to obtain the distribution as well as ROC curves for speech and music using LEF, MLER and RSF features, respectively.

A block diagram is shown in figure 4.1 to illustrate the process of extracting the RSF feature from one track of audio. This may not be the most efficient way of implementing this algorithm of RSF feature extraction, but it demonstrates one way of doing it. For instance, this structure computes the S parameter for all frames in parallel. This might be essential for real-time applications to speed up the feature extraction process. However, for data retrieval, where time is not crucial, serial processing of frames might be adequate.

Two databases were used to examine the three features:

1. Speech database provided by University College London (UCL) [40]
2. Music database provided by Real World Computing (RWC) [41-43]

Both sets of data were recorded in an anechoic chamber and sampled at 44.1 kHz.

Figures 4.2, 4.3 and 4.4 show the distribution of LEF, MLER, and RSF features, respectively. They were obtained for speech and music signals when analyzing 3000 speech tracks (1 second each) and similar number of music tracks. Each audio track was split into 20 ms frames (M = 50 frames, R = 44100/50 = 882 samples).

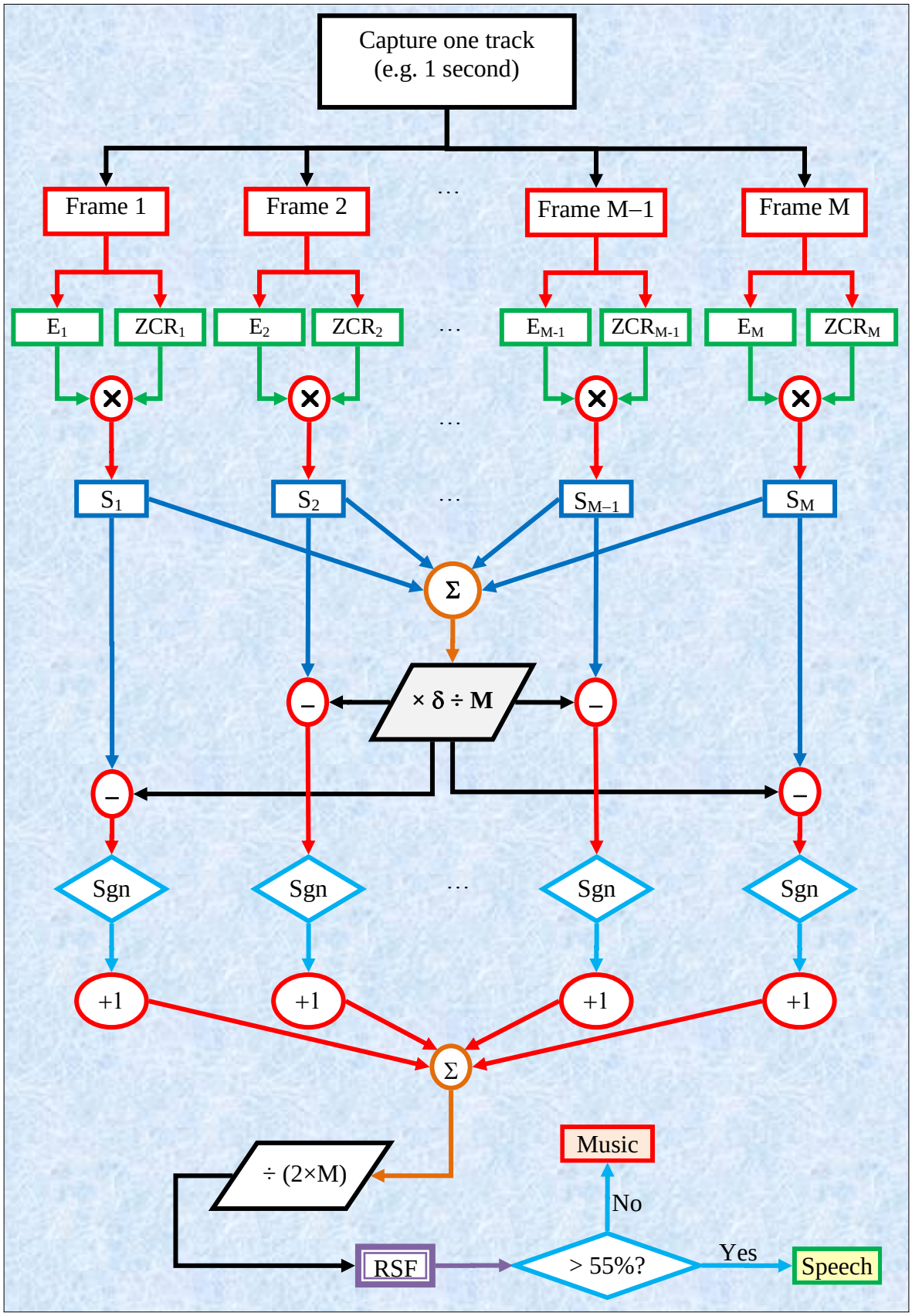


Figure 4.1 Block diagram for the RSF feature extraction from an audio track

As can be seen from the speech and music distributions in figures 4.2, 4.3 and 4.4 there is an overlap that implies certain misclassification error. For example, if a threshold is used to discriminate between speech and music (e.g. at LEF = 50%) then it can be found that 86% of speech tracks are above this threshold. Thus, 86% of speech signals will be correctly classified as speech. At the same time there are also music tracks above the same threshold. Using the same threshold, it can be found that 22% of music tracks will be misclassified as speech. There is always a trade-off between correct classification and misclassification.

Choosing an optimum threshold is the subject of the following section. The same procedure was repeated for the MLER and RSF features using $\delta = 0.1$ as recommended in [23]. The following section investigates the method adopted to assess the three features and to compare them to each other.

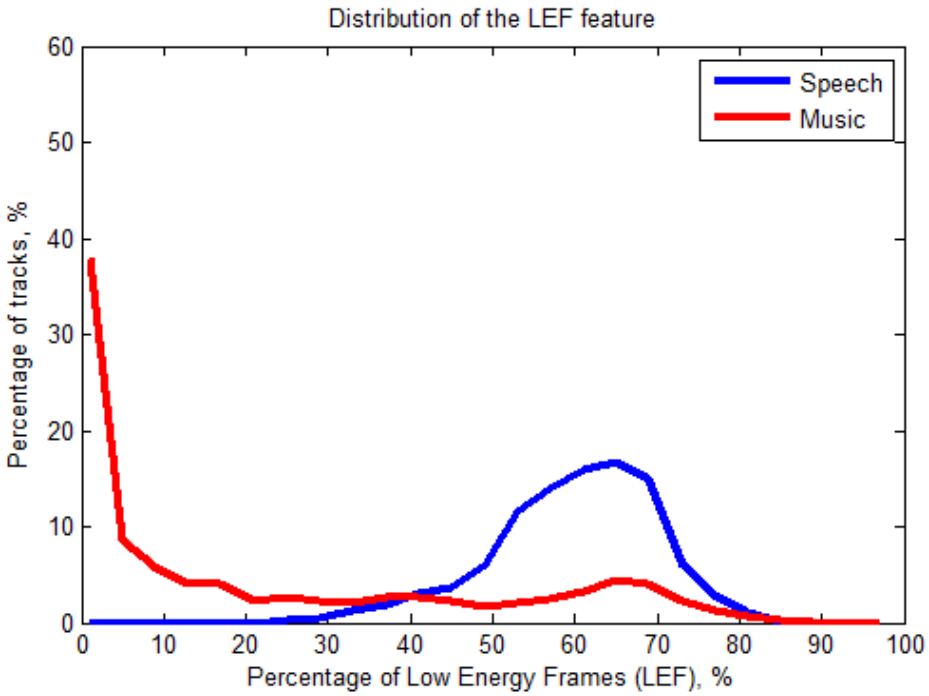


Figure 4.2 Distribution of the LEF feature for speech and music

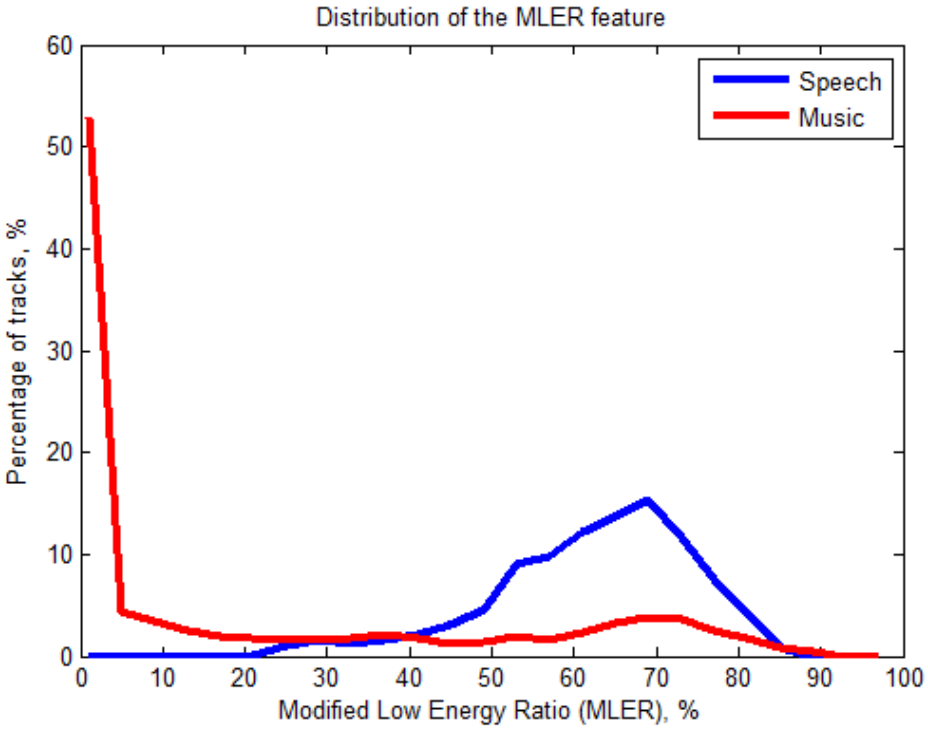


Figure 4.3 Distribution of the MLER feature for speech and music

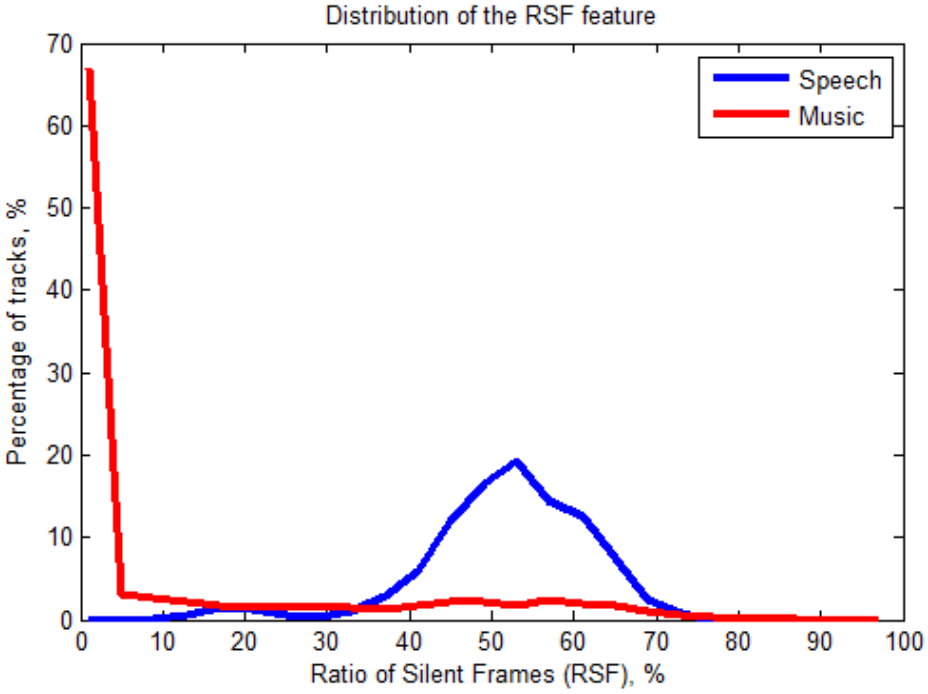


Figure 4.4 Distribution of the RSF feature for speech and music

4.5. Comparison by ROC Curves

The ROC curves can be used to compare the performance of the three types of features and to find the optimum threshold to be used for speech/music detection. The ROC curves of the LEF, MLER, and RSF features are shown in figures 4.5, 4.6, and 4.7, respectively. Choosing a particular True Positive (TP) and False Positive (FP) value for either speech or music detection is a choice of design. A designer might choose a point on the curve where high true positive detection of speech is obtained, while risking high false positive. However, for the purpose of comparing the three features discussed in this chapter, the optimum point on the ROC curve for each feature was computed where the Euclidian distance to the ideal point, i.e. (FP, TP) = (0,1), is minimal; i.e. point D on figure 3.6.

As described in chapter 3, the Euclidian distance between any point on an the ROC curve, i.e. (FP, TP) = (x,y), and the ideal point D, i.e. (FP, TP) = (0,1), is computed using:

$$\text{Euclidian Distance} = \sqrt{[(x-0)^2 + (y-1)^2]} \tag{4.13}$$

It can be observed from figures 4.5, 4.6, and 4.7 that speech ROC curves are different from music ROC curves. This means a music detection system is different from a speech detection system. For that reason, the optimum threshold for speech detection is different from the optimum threshold for music detection. Table 4.1 shows the optimum TP rates and FP rates for speech and music detection when using LEF, MLER, and RSF features. It also summarizes the enhancement introduced by MLER and RSF features. An enhancement is achieved when the TP rate is increased and the FP rate is decreased.

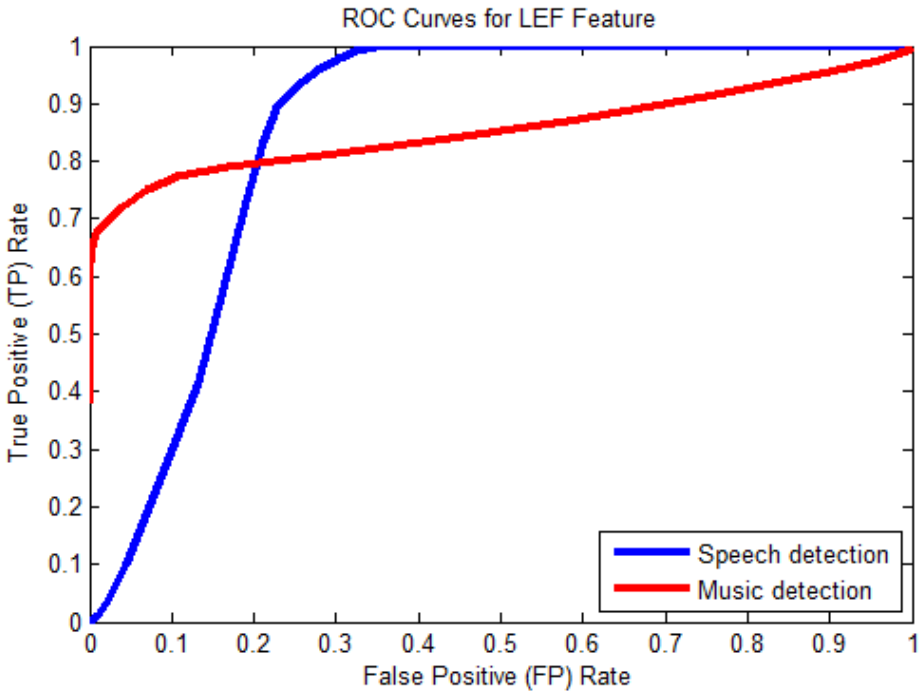


Figure 4.5 ROC curves for speech/music detection when using the LEF feature

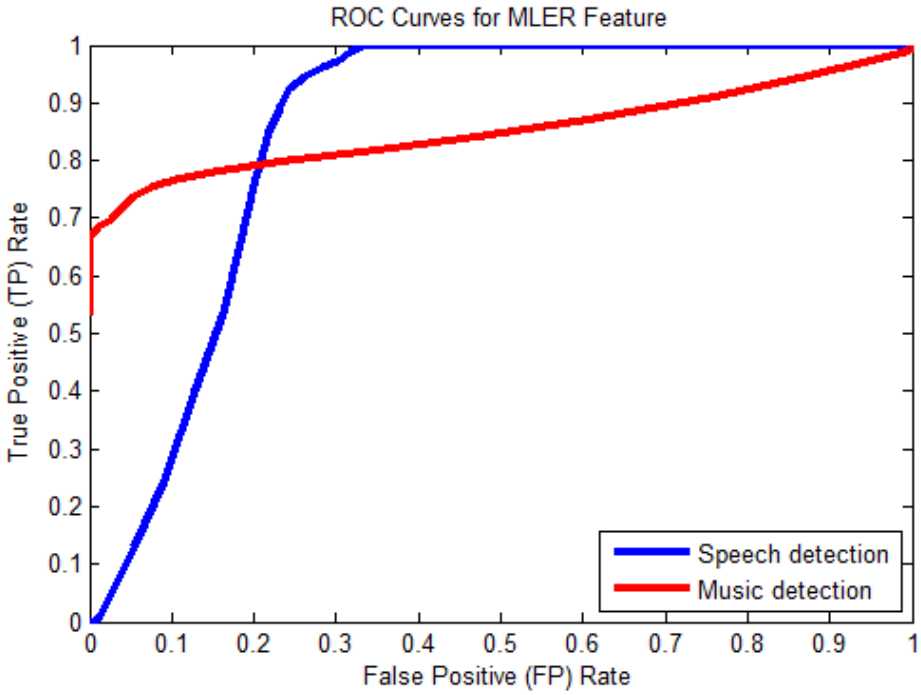


Figure 4.6 ROC curves for speech/music detection when using the MLER feature

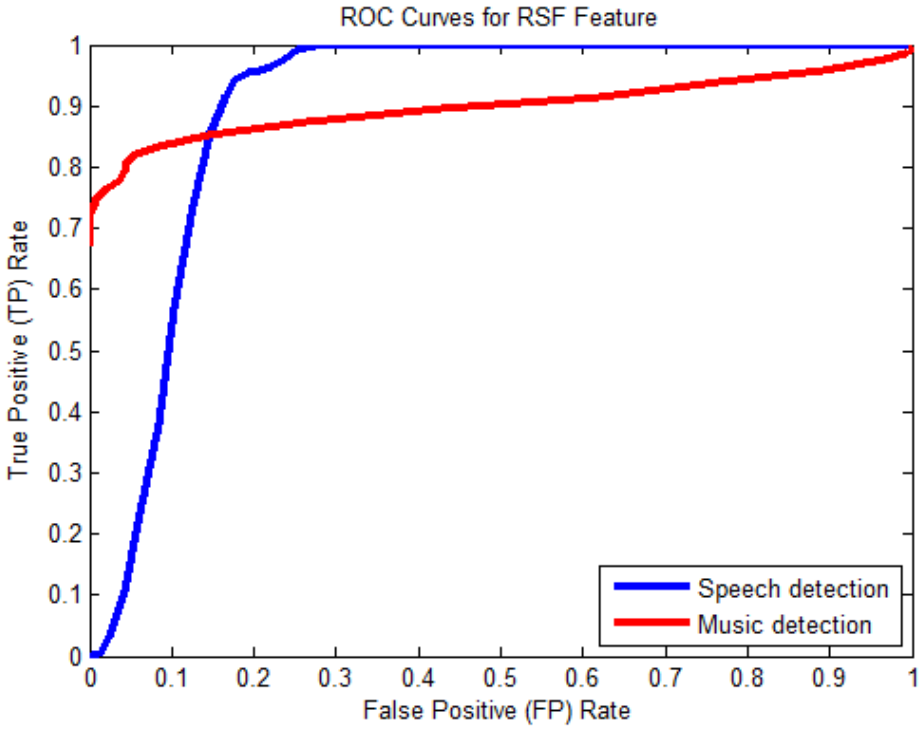


Figure 4.7 ROC curves for speech/music detection when using the RSF feature

	TP		FP	
	Speech	Music	Speech	Music
LEF	0.893	0.773	0.227	0.107
MLER	0.894	0.768	0.232	0.106
RSF	0.915	0.835	0.165	0.085
MLER Enhancement over LEF	0.1%	–0.7%	2%	–1%
RSF Enhancement over LEF	2.4%	8%	–27%	–20%
RSF Enhancement over MLER	2.3%	8.7%	–29%	–19.6%

Table 4.1 True Positive (TP) rates and False Positive (FP) rates for LEF, MLER, and RSF

To summarize, the product of the frame's energy and its ZCR was utilized to label silent frames. The ratio of silent frames (RSF) in a 1-second track was used as a feature for speech/music discrimination. The RSF feature enhanced the speech/music discrimination significantly over the LEF and MLER features. This was verified experimentally.

As for the RSF feature of the examined datasets, the following conclusionscan be extracted from table 6.1:

1. Speech detection:
    - TP rate of RSF feature is 2.4% higher than LEF
    - TP rate of RSF feature is 2.3% higher than MLER
    - FP rate of RSF feature is 27% lower than LEF
    - FP rate of RSF feature is 29% lower than MLER
2. Music detection:
    - TP rate of RSF feature is 8% higher than LEF
    - TP rate of RSF feature is 8.7% higher than MLER
    - FP rate of RSF feature is 29% lower than LEF
    - FP rate of RSF feature is 19.6% lower than MLER

4.6. Summary

In this chapter a novel feature called the Ratio of Silent Frames (RSF) has been presented and analysed. The feature was compared to two other features from the literature: the Low Energy Frames (LEF) feature and the Modified Low Energy Ratio (MLER) feature. The RSF feature was found to provide significant enhancement over other relevant features. The following chapter introduces another novel feature that has been shown to perform better than the RSF feature.

Chapter 5: Time Series Events (TSE)

5.1. Introduction

In the previous chapter, a novel feature was introduced showing certain enhancement to two relevant features from the literature. In this chapter, another novel feature is introduced showing far better performance in discriminating between speech and music.

Let us start by revisiting a very fundamental feature for audio signals: the zero crossing rate (ZCR) [16]. It has been widely used in the literature to discriminate between speech and music [1, 21, 23, 30, 31] and can be computed using the script shown in listing A11. However, its compliment feature (Non-ZCR) has not been investigated [11, 12]. This chapter provides a novel contribution to this field of research by:

1. Defining 9 types of Time Series Events [11] and investigating their probability of occurrence in an audio signal; i.e. speech and music,
2. Introducing a novel speech/music discrimination feature: the probability of the occurrence of the Non Zero Crossing Events, and
3. Applying Receiver Operating Characteristics (ROC) curves and using them to assess the novel speech/music discrimination feature [12].

5.2. Time Series Events

A digital signal is composed of a series of samples that vary in time. This variation makes each sample in the signal fall in one of three regions: Positive amplitude (+), null amplitude (0), or negative amplitude (–). Figure 5.1 illustrates this by an example which shows:

1. Samples at n = 1, 2, 4, 7, 11, and 12 are above zero (+),
2. Samples at n = 5, 8, and 9 are null (0),
3. Samples at n = 3, 6, and 10 are below zero (–).

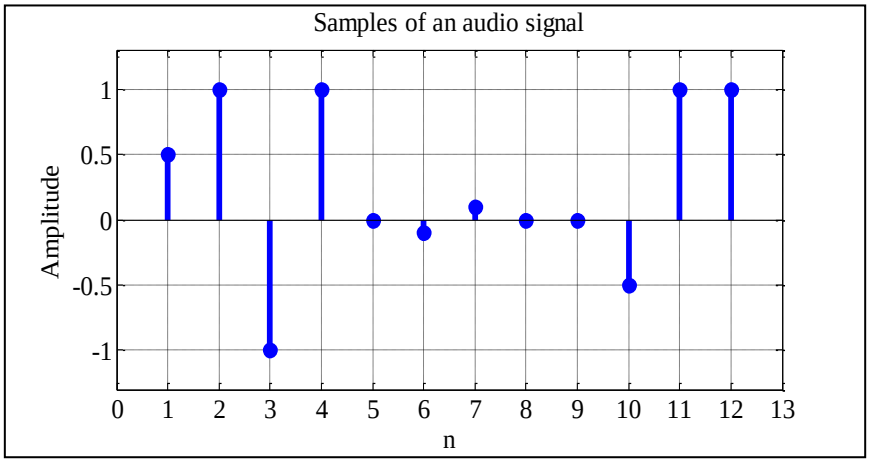


Figure 5.1 A snapshot of a series of samples taken from an audio signal

Any two consequent samples in a time series can form an event; i.e. a time series event. For instance, a zero crossing event occurs when a positive sample is followed by a negative sample (e.g. samples at n = 2 and n = 3), or vice versa (e.g. samples at n = 10 and n = 11). The zero crossing event is not the only possible event that could occur in any digital signal. In fact there are 9 possible types of events. These types are listed in table 5.1. An example is shown for each type.Every type of event in table 5.1 needs to be investigated

The probability of the occurrence of each event for any signal can be found using the following fundamental equation:

$$\text{P(event } \mathbf{E}) = \frac{\text{Numberoftimesevent } \mathbf{E} \text{ occured}}{\text{Total numberofevents}} \qquad (5.1)$$

$$\text{Total number of events} = N - 1 \quad \text{, where N is the number of samples} \qquad (5.2)$$

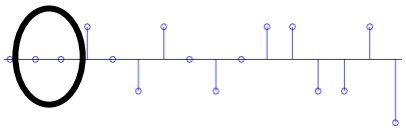
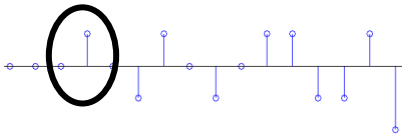
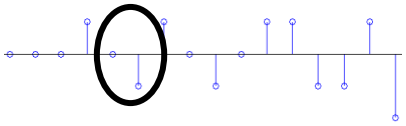
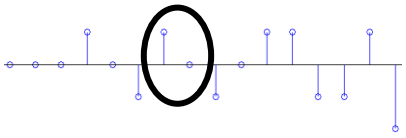
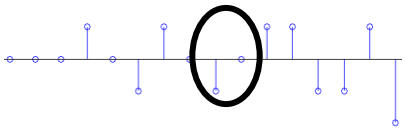
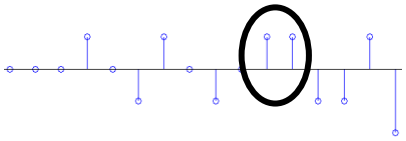
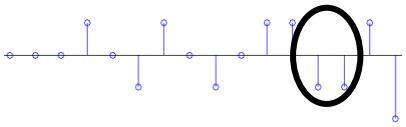
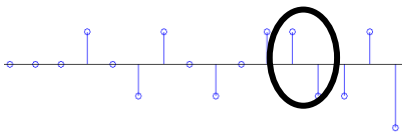
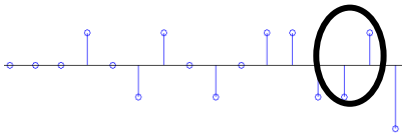
Event No.	Consequent samples		Example	Name of Event (If applicable)
Event 1	0	0		
Event 2	0	+		
Event 3	0	–		
Event 4	+	0		
Event 5	–	0		
Event 6	+	+		Non Zero Crossing Event ($NZCE_{++}$)
Event 7	–	–		Non Zero Crossing Event ($NZCE_{--}$)
Event 8	+	–		Zero Crossing Event (ZCE_{+-})
Event 9	–	+		Zero Crossing Event (ZCE_{-+})

Table 5.1 All possible events that could occur in a time series of a sampled signal

5.3. Probabilities of Events

In order to compute the probability of the occurrence of every type of event in table 5.1, equation 5.1 was applied on two audio tracks: a speech track and a music track. The code scripted to carry out this computation is shown in listing A12. Each track was 3 seconds long. The probability of the nine various events are illustrated by figure 5.2. The track length was arbitrarily chosen to be 3 seconds. However, its effect on the discrimination performance will be investigated later in chapter 6.

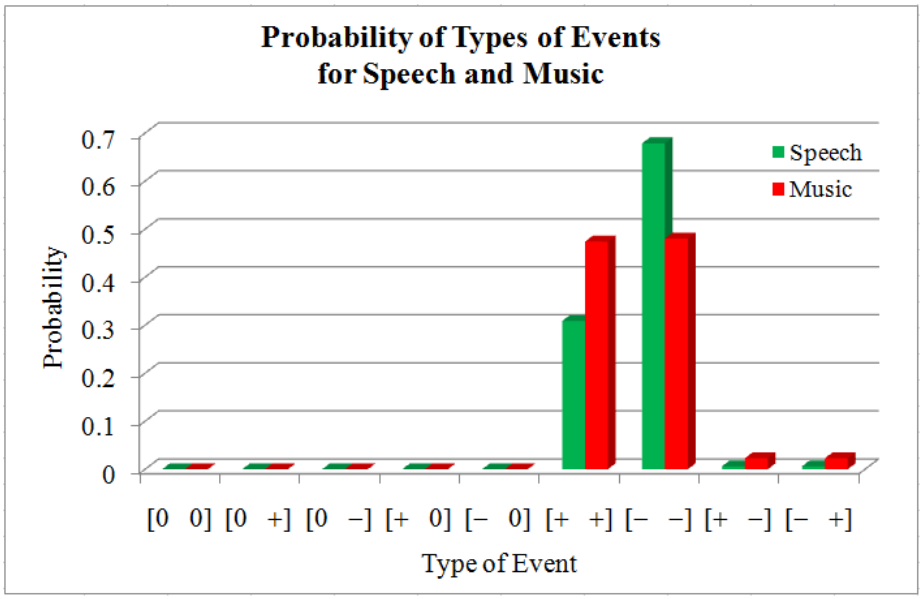


Figure 5.2 The probability of occurrence for every type of event in a 3 seconds speech track and a 3 seconds music track, sampled at f_s = 44.1 kHz

The following observations can be noticed from figure 5.2:

1. The first 5 types of events have very low probability of occurrence; i.e. almost zero. In general, zero amplitude is seldom recorded. Instead, it would be very low amplitude in the vicinity of zero.

2. The Non Zero Crossing Events ($NZCE_{++}$and $NZCE_{--}$) show relatively high probability of occurrence and significant discrepancy between the two types of signals. Specifically, music has a probability of around 0.45 for both types of events. However, speech has a probability of 0.3 for $NZCE_{++}$and 0.65 for $NZCE_{--}$. The latter discrepancy is a property that could be exploited to discriminate between speech and music.
3. The Zero Crossing Events ($ZCE_{+-}$and ZCE_{-+}) show relatively low probability of occurrence with some discrepancy between the two types of signals.

5.4. Distributions

The results shown in the previous section may not necessarily be true for every speech and music track. The validity of any scientific experiment depends on its repeatability. The same experiment should be repeated for a large number of times and the distributions of the outcome of the experiment are then considered. For such statistical validity, 3000 speech tracks and 3000 music tracks were tested. Each track was 3 seconds long.

The total time of the examined data was 9000 seconds; i.e. 150 minutes of speech and 150 minutes of music (totalling 5 hours). For each track the probability of every significant event explained in the previous section was analysed and will be discussed in the following subsections.

5.4.1. Non zero crossing events (NZCE)

Figures 4.3 and 4.4 show the distributions of the probability of the two types of NonZero Crossing Events ($NZCE_{++}$) and ($NZCE_{--}$), respectively. The probability of the occurrence of

each type of the NZCE's could be either computed using listing A12 or listing A13. In order to get the distribution of the probability of these two events, a stream of sound was collected from the dataset and split into equal tracks (3 seconds each), removing any long silence, and computing the probability of any examined event to occur as shown in listing A14. When repeating this process for 3000 tracks as in listing A15, a distribution was obtained characterizing the events. It is evident that this feature has insignificant overlap (green shaded area) between the speech and music distributions; i.e. the likelihood of misclassifying speech as music, or vice versa, is very small.

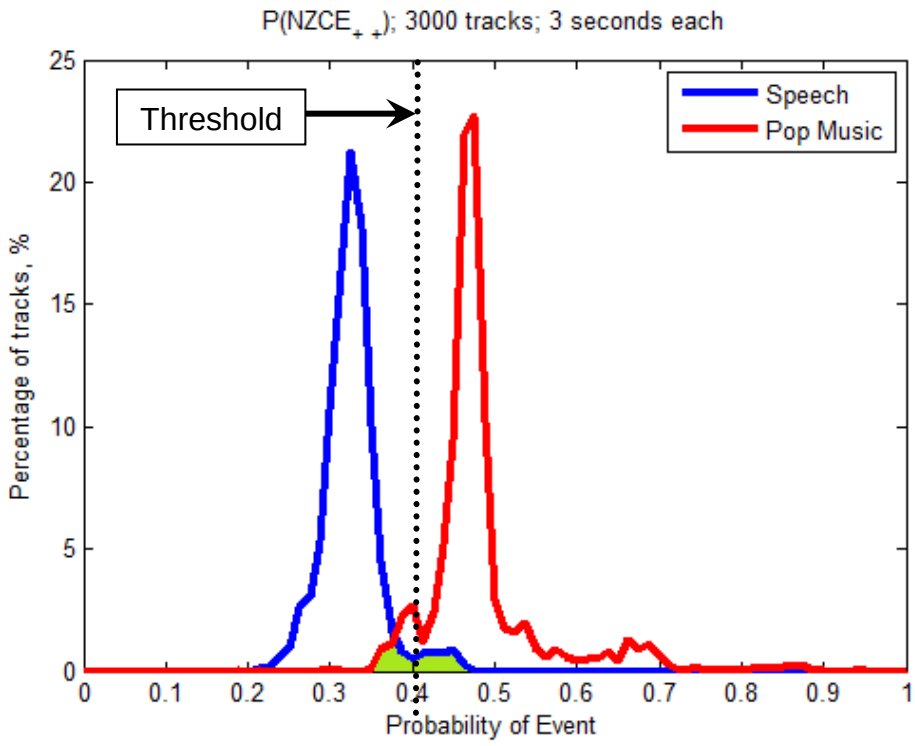


Figure 5.3 Distribution of the $P(NZCE_{++})$ for speech and music when examining 3000 speech tracks and 3000 music tracks each 3 seconds long, sampled at f_s = 44.1 kHz

By visual inspection of the two distributions in figure 5.3, it can be observed that setting a threshold at the probability of 0.4 would provide an acceptable discrimination between speech and music whereas a threshold of 0.55 would be more appropriate in figure 5.4. Nevertheless, determining the optimum threshold can also be obtained using a Receiver Operating Characteristics (ROC) curve (section 3.5).

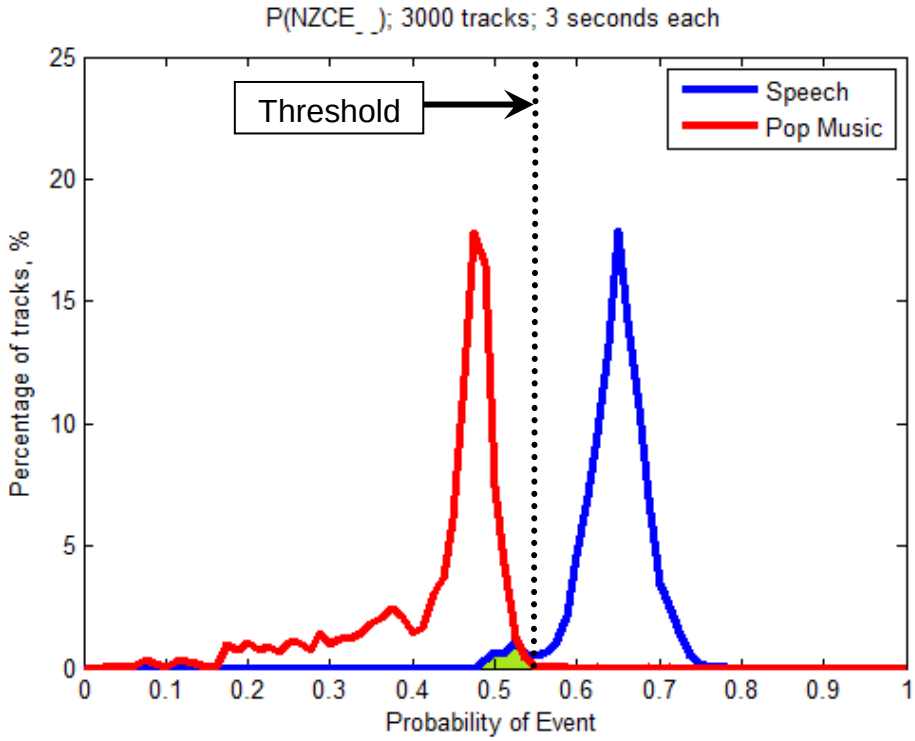


Figure 5.4 Distribution of the P(NZCE$_{--}$) for speech and music when examining 3000 speech tracks and 3000 music tracks each 3 seconds long, sampled at f_s = 44.1 kHz

5.4.2. Zero crossing events (ZCE)

Figure 5.5 shows the distribution of the probability of ZCE$_{+-}$. A similar distribution was obtained for the other type of ZCE$_{-+}$. Listing A15 was also used to obtain this distribution.

It can be observed that the amount of overlap between speech and music tracks is relatively large (green shaded area) when compared to any of the NZCE types.

Due to such obvious overlap, every type of ZCE seems to perform much worse than any type of NZCE. In spite of the overlap between the speech and music distributions, the Zero Crossing Rate (ZCR) feature is still a useful characteristic that was exploited many times in the literature [16, 31, 32]. It is normally used in conjunction with other features to form a vector of parameters for the purpose of discriminating between speech and music. However, when used alone it does not seem to be as good. This feature was exploited in chapter 4 for the purpose of detecting silent frames in an audio track in order to discriminate between speech and music.

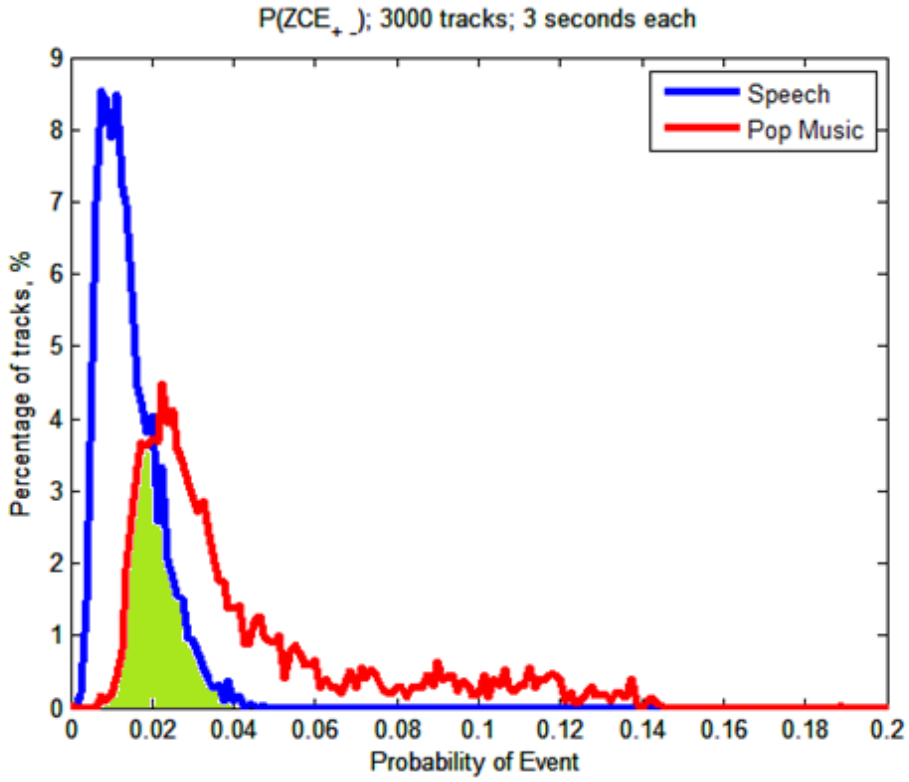


Figure 5.5 Distribution of the P(ZCE$_{+-}$) for speech and music when examining 3000 speech tracks and 3000 music tracks each 3 seconds long, sampled at f_s = 44.1 kHz

5.5. Sets of Events

The individual events discussed in the previous sections can be grouped into sets that share similar definitions. Figure 5.6 shows a Venn diagram that illustrates the space of events which contains three major sets. Comparing the distributions of these sets of events would help in explaining why the complete NZCE set does not have a significant advantage over the ZCE set.

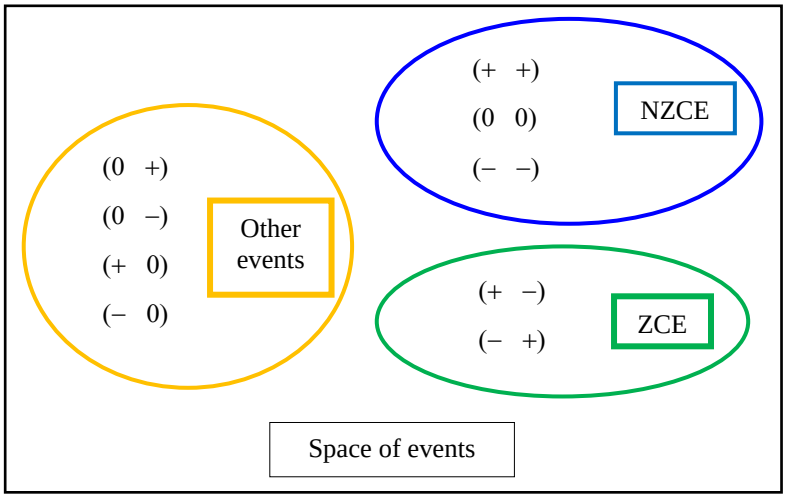


Figure 5.6 Venn diagram for the sets of events and their elements

In the following subsections, every set is treated as a single type of event. For example, when computing the probability of ZCE, every event of the type (+ −) or (− +) is counted as an occurrence of ZCE. Splitting the whole space into just two broad categories (i.e. ZCE and NZCE) makes them a compliment of one another. This makes the two distributions to be mirror images of each other. The sets would seem to hide the useful information that are provided by their elements; i.e. individual events.

For any set of two possible events,

$$P(Event\ A \cup Event\ B) = P(Event\ A) + P(Event\ B) - P(Event\ A \cap Event\ B) \quad (5.3)$$

Every two events among the 9 possible events are considered to be disjoint. In other words, if one event occurs in a pair of consequent samples, the possibility of another type of event to occur at the same time is null. Mathematically this can be expressed as:

$$P(Event\ A \cap Event\ B) = \phi \tag{5.4}$$

Substituting equation 5.4 into equation 5.3 results in equation 5.5:

$$P(Event\ A \cup Event\ B) = P(Event\ A) + P(Event\ B) \tag{5.5}$$

This explains why a set would mask the information provided by an individual type of event. The ZCE of both types in the set [(+ –), (– +)] had low probability; i.e. in the range [0 to 0.15]. Their summation is expected to be in the same vicinity. On the other hand, the NZCE of both types [(+ +), (––)] had relatively much higher probabilities summing up to be closer to 1. In order to verify this, the peak of each distribution was used to examine equation 5.5 for the NZCE case.

Table 5.2 shows the results of summing the peak of the speech and music distributions for NZCE for all three types (+ +,– –, and 0 0). The summation of the individual probabilities is equal to the probability of the NZCE set.

The set named “other events” in the Venn diagram has close to zero probability of occurrence. In fact, in some audio signals, they never occurred at all. Therefore, they carry no information that is worth analysing in the context of speech/music discrimination.

Event Type	Peak Probability of Event Occurrence	
	Speech	Music
$NZCE_{++}$	0.320	0.470
$NZCE_{--}$	0.645	0.475
$NZCE_{00}$	0.010	0.010
Σ Probabilities	0.975	0.955
NZCE set	0.975	0.955

Table 5.2 Summation of the peak of distributions for both elements in the NZCE set

5.5.1. ZCE set

The union of the two types of ZCE features: ZCE_{+-} and ZCE_{-+} is referred to here as the ZCE set. Computing the probability of this set to occur implies counting every type of ZCE regardless of which type it is. After computing the probability of the ZCE set to occur in every track for 3000 tracks, a distribution can be obtained to analyse the distributions for speech and music signals. The distribution that results from this set is shown in figure 5.7. The overlap between the speech and music distributions is relatively much more significant than any of its constituent elements.

When comparing figure 5.5 and figure 5.7 we can see that they are different. This is because figure 5.5 shows the distribution of one type of ZCE; i.e. ZCE_{+-}. The other figure, figure 5.7, shows the distribution for the whole set of ZCE types; i.e. summation of ZCE_{+-} and ZCE_{-+}.

5.5.2. NZCE set

The union of the two types of NZCE features: $NZCE_{++}$ and $NZCE_{--}$ here is referred to as the NZCE set. Like the ZCE set, the NZCE set suffers from significant overlap between the speech and music distributions. It is evident from figure 5.7 and figure 5.8 that the ZCE and the NZCE sets are mirror images of each other. This is expected since the total space of events is composed of these two main categories: NZCE set and ZCE set; see figure 5.6. It seems that the events that are useful in discriminating between speech and music are masked by the sets they belong to.

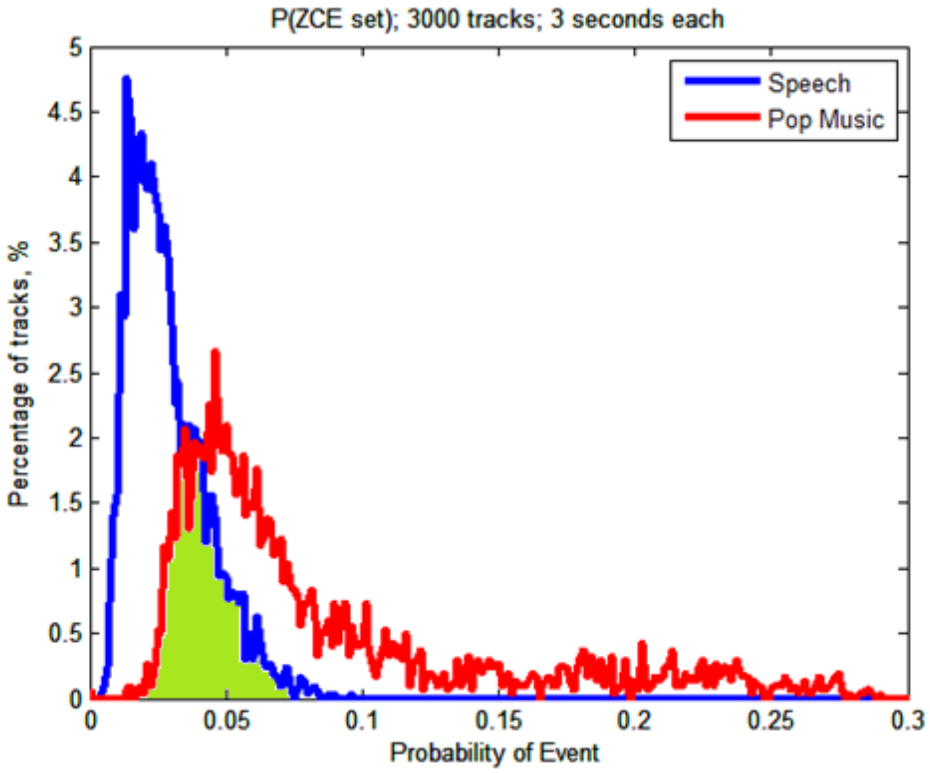


Figure 5.7 Distribution of the P(ZCE set) for speech and music tracks

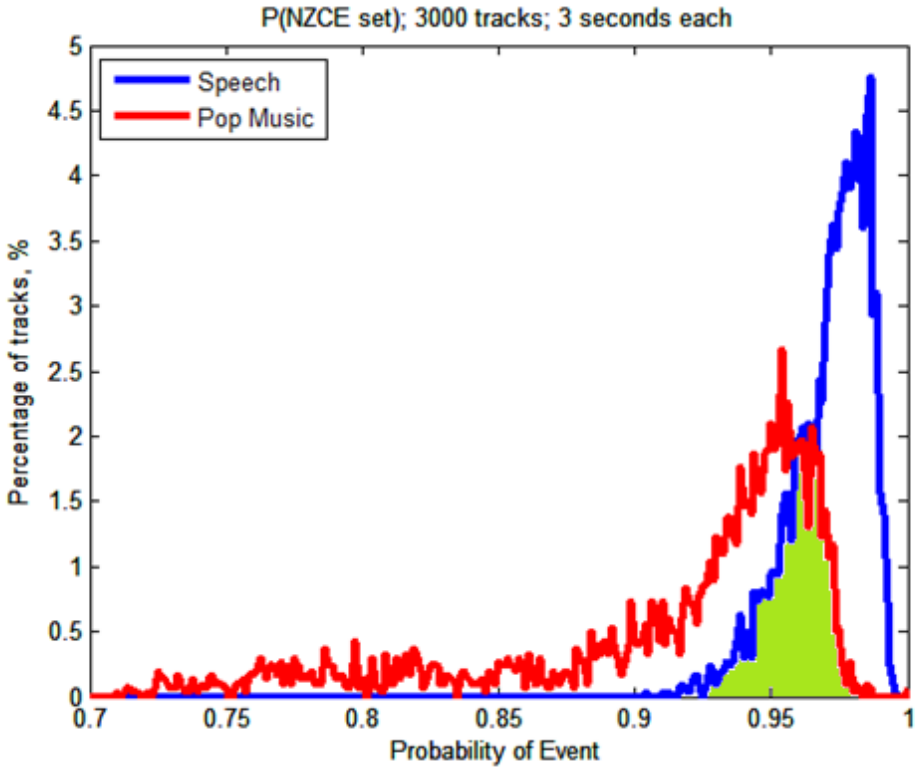


Figure 5.8 Distribution of the P(NZCE set) for speech and musictracks

5.6. Mathematical Model

The investigation carried out so far implies that the events of type (+ +) and (– –) are the best events to be used in discriminating between speech and music signals. The probability of either event can be computed using equation 5.6 below:

$$\mathrm{P(NZCE)} = \frac{\sum_{n=1}^{N-1}[Sign\,[S(n)].Sign\,[S(n+1)]]}{N-1} \tag{5.6}$$

Where: *N* is the number of samples in the signal,

Sign (*X*) is defined based on the type of the NZCE as follows:

Case 1: NZCE $_{++}$

$$Sign(X) = \begin{cases} 1 & X > 0 \\ 0 & X = 0 \\ 0 & X < 0 \end{cases} \quad (5.7)$$

Case 2: NZCE $_{--}$

$$Sign(X) = \begin{cases} 0 & X > 0 \\ 0 & X = 0 \\ 1 & X < 0 \end{cases} \quad (5.8)$$

Using such a mathematical expression to compute the probability of the NZCE makes their simulation or implementation faster than counting events one by one.

5.7. ROC Curves for Time Series Events (TSE)

The ROC curves can be used to achieve two goals: i) compare the performance of the various types of events and ii) find the optimum threshold to be used for speech/music detection. Consider the example of the radio receiver that scans for news channels by trying to detect detection of speech. Choosing between high probability of correct detection of news (speech) while also risking higher probability of false alarms (music) is a choice of design. A designer might choose a point on the curve where high true positive detection of speech is obtained, while risking high false positive. Nevertheless, it is recommended to choose a point on the curve where the Euclidian distance to the ideal point is minimal; i.e. point D on figure 3.6 where (FP, TP) = (0,1).The Euclidian distance between any point on a ROC curve, e.g. (FP, TP) = (x,y), and the ideal point D, i.e. at (FP, TP) = (0,1), is computed using:

$$\text{Euclidian Distance} = \sqrt{[(x-0)^2 + (y-1)^2]} \quad (5.9)$$

This is computed for the four significant individual events as well as the two sets of events they belong to. Another useful figure of merit is the area under the ROC curve which ranges from 0 to 1. The ideal curve illustrated in figure 3.6 has got the area 1 under its curve; i.e. the area of the 1 × 1 square. On the other hand, the random performance (which contains the point C) curve has an area of 0.5. Thus, the area under the ROC curve is another useful comparative parameter that can be used to compare any two features.

Figures 5.9and5.10 show the ROC curves for the two types of NZCE: (+ +) and (– –). It is evident that they are very close to the ideal curve. The two types of NZCE are not the same, and they are expected to be different; their distributions are different as demonstrated in figures 5.3 and 5.4. In fact, the distributions of the two types are mirror images of each other and are compliment of each other within the same NZCE set. In general, their performance is above 95%; i.e. optimum TP rate > 0.95.On each ROC curve, the optimum TP rate and FP rate, the optimum threshold, and the area under each ROC curve for both speech and music detection cases are also summarized. There is always slight difference between the optimum threshold for detecting speech and that for detecting music. This is expected since the distributions are not symmetrical.

Figure 5.11 illustrates the ROC curves for detecting speech and music using the $P(ZCE_{+-})$. Similar ROC curves were obtained for the $P(ZCE_{-+})$. For both types, the optimum TP rate is about 0.79 and 0.82 for speech and music detection, respectively. In general, about 80% performance is achievable for every type of the ZCE features in comparison to more than 95% in case of NZCE types of events. This implies that the NZCE features improve the true positive (TP) detection by 15% compared to the ZCE features.

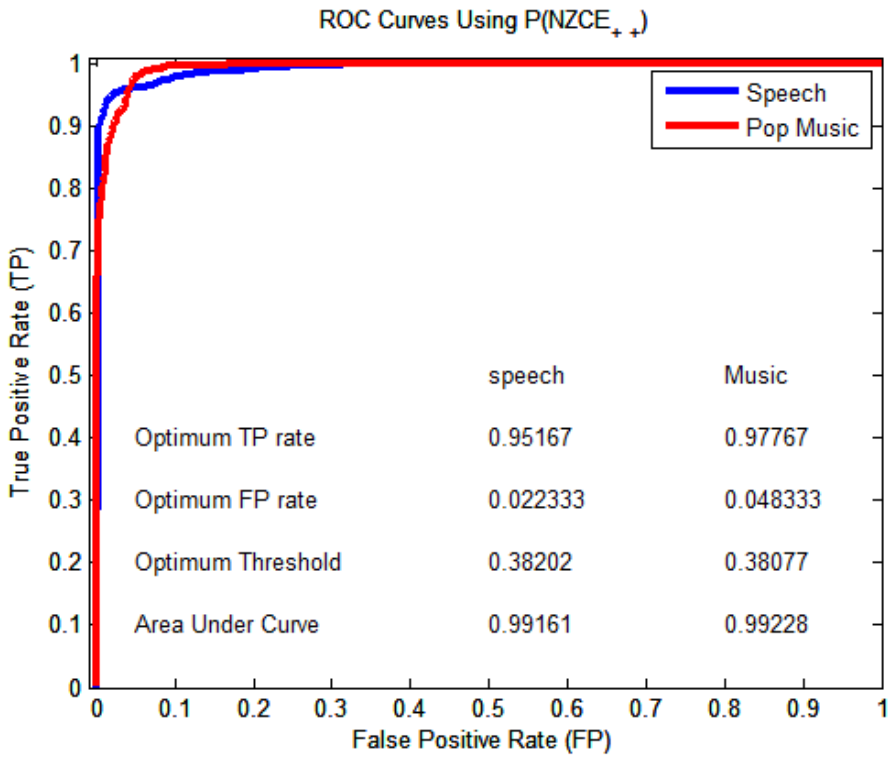


Figure 5.9 ROC curves for using P($NZCE_{+\,+}$) to analyse 3000 tracks, 3 seconds each

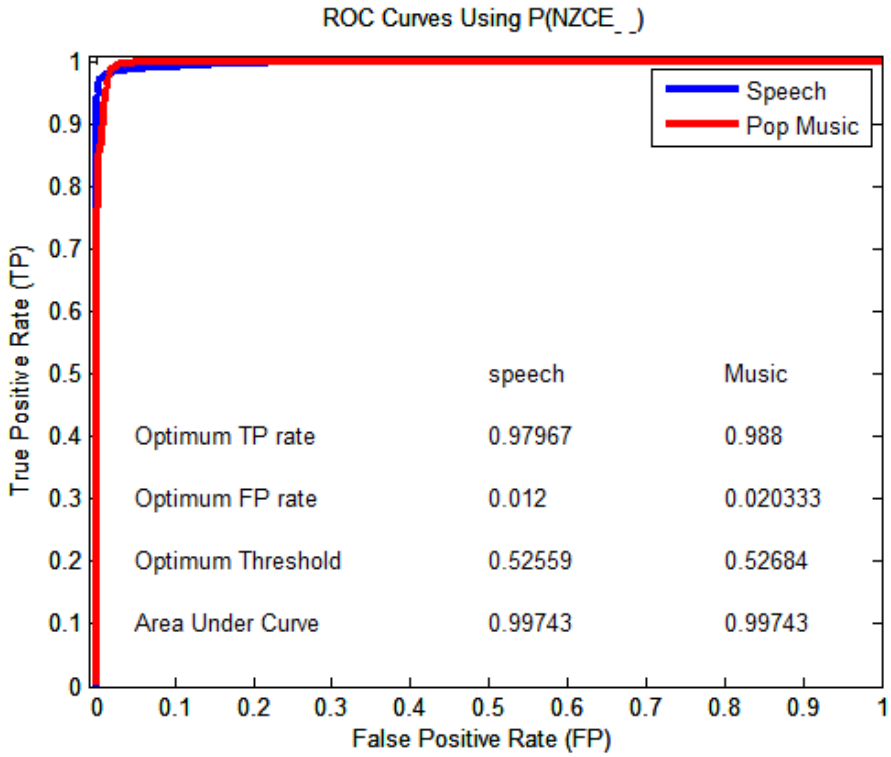


Figure 5.10 ROC curves for using P($NZCE_{-\,-}$) to analyse 3000 tracks, 3 seconds each

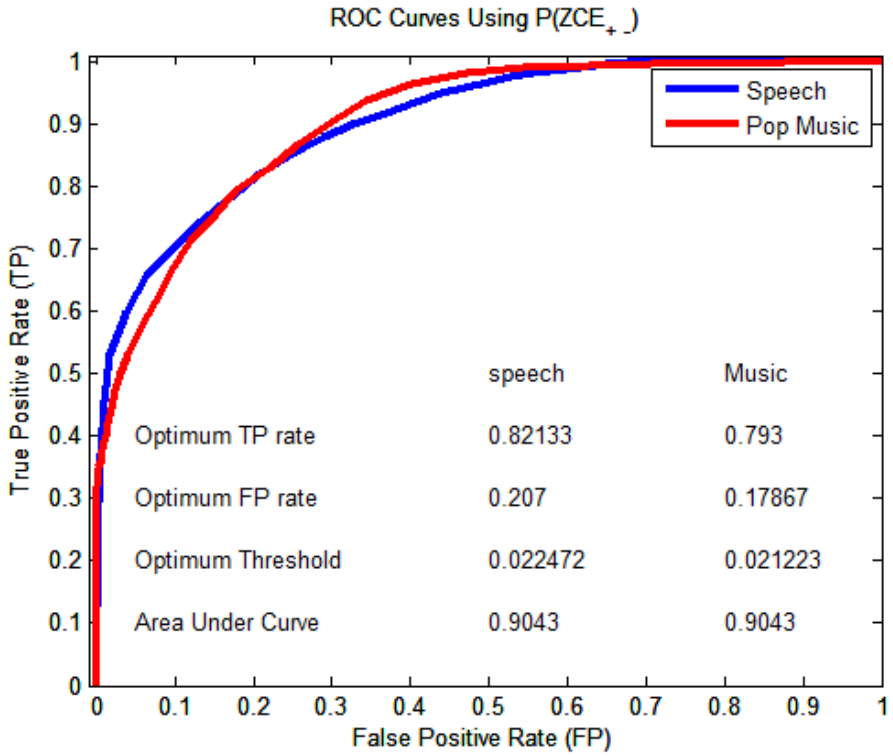


Figure 5.11 ROC curves for using $P(ZCE_{+-})$ to analyse 3000 tracks, 3 seconds each

Figures 5.12 and 5.13 show the ROC curves for the two main sets in the space of events: NZCE set and ZCE set, respectively. When comparing these curves to the ZCE individual types of events, i.e. $P(ZCE_{+-})$ and $P(ZCE_{-+})$, there is no significant difference in performance. The four of them are far from the ideal curve. For the examined datasets, tables 5.3 and 5.4 summarize the performance parameters of the ROC curves for detecting speech and music, respectively.

In order to compare the six features that have been examined, all ROC curves are plotted on the same graph in figure 5.14. The superiority of the two individual types of NZCE features is very obvious. It is also evident that both of the individual ZCE features as well as the set of ZCE and set of NZCE features all coincide with each other giving the lowest performance.

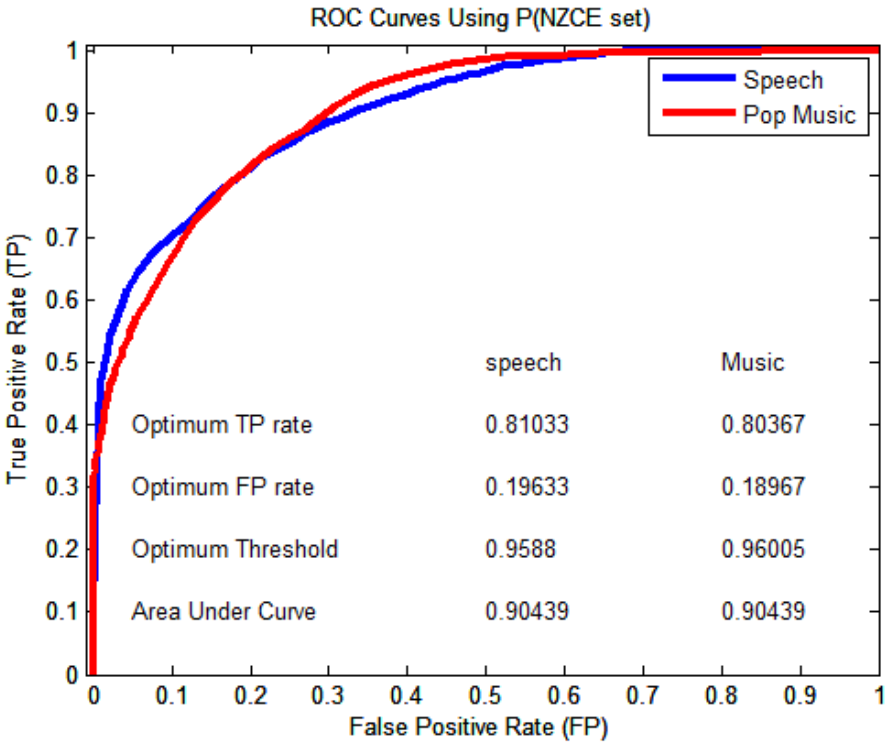


Figure 5.12 ROC curves for using P(NZCE set) to analyse 3000 tracks, 3 seconds each

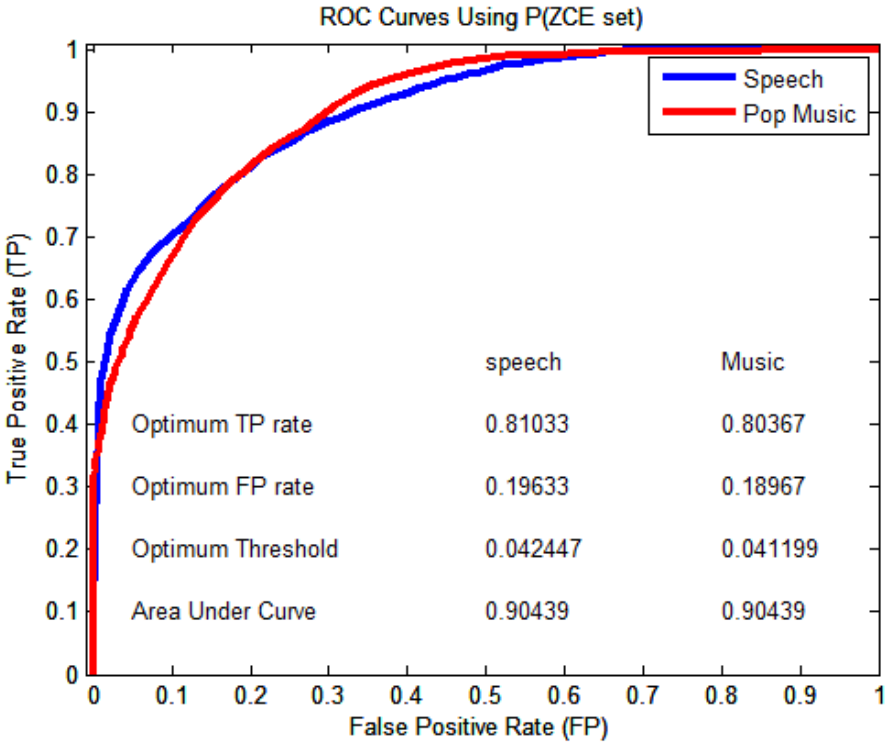


Figure 5.13 ROC curves for using P(NZCE set) to analyse 3000 tracks, 3 seconds each

Parameter / Feature	Optimum TP rate	Optimum FP rate	Optimum Threshold	Euclidian Distance[1]	Area under ROC curve
NZCE $_{++}$	0.9517	0.0483	0.3820	0.0532	0.9916
NZCE $_{--}$	0.9797	0.0203	0.5256	0.0236	0.9974
ZCE $_{+-}$	0.8213	0.1787	0.0225	0.2734	0.9043
ZCE $_{-+}$	0.8217	0.1783	0.0225	0.2732	0.9043
NZCE set	0.8103	0.1897	0.9588	0.2730	0.9044
ZCE set	0.8103	0.1897	0.0424	0.2730	0.9044

Table 5.3 Summary of ROC curve parameters of examined features for speech detection

Parameter / Feature	Optimum TP rate	Optimum FP rate	Optimum Threshold	Euclidian Distance[1]	Area under ROC curve
NZCE $_{++}$	0.9777	0.0483	0.3808	0.0532	0.9923
NZCE $_{--}$	0.9880	0.0203	0.5268	0.0236	0.9974
ZCE $_{+-}$	0.7930	0.1787	0.0212	0.2734	0.9043
ZCE $_{-+}$	0.7930	0.1783	0.0212	0.2732	0.9043
NZCE set	0.8037	0.1897	0.9600	0.2730	0.9044
ZCE set	0.8037	0.1897	0.0412	0.2730	0.9044

Table 5.4 Summary of ROC curve parameters for examined features when detecting music

[1]Euclidean distance is computed at the optimum TP and FP rates

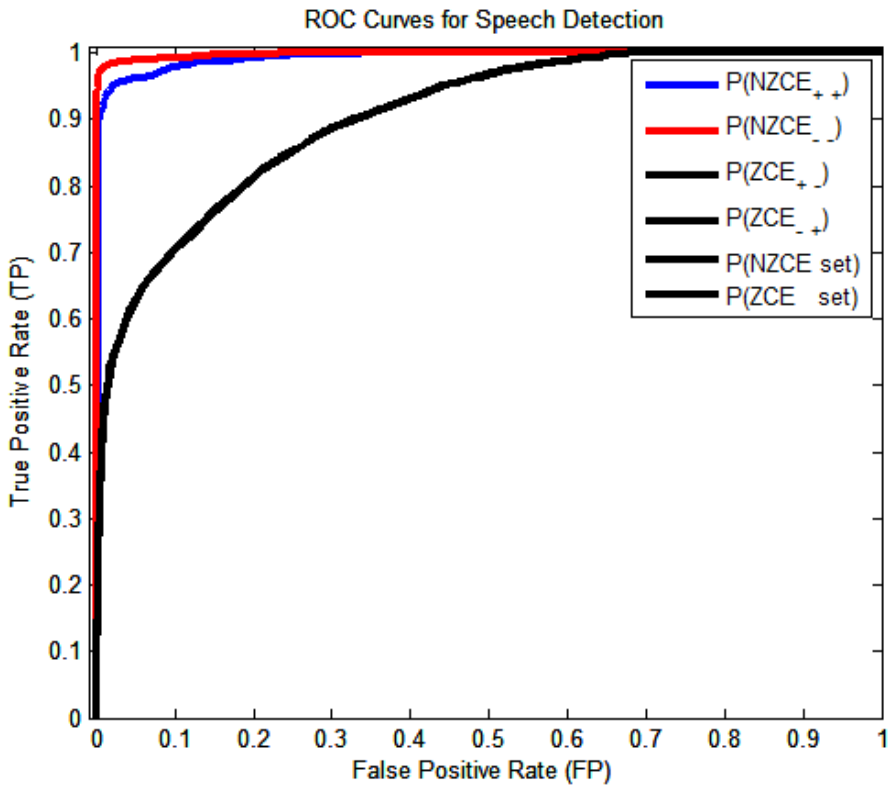


Figure 5.14 ROC curves for all examined features plotted for comparison

5.8. Delta P(NZCE)

By observing figure 5.2, it can be noticed that there is a significant discrepancy between the $P(NZCE_{+\,+})$ and the $P(NZCE_{-\,-})$ for the speech track. However, there is no such discrepancy between the two features for the music track; i.e. they are equal. Such discrepancy between the two types of NZCE features could be exploited by investigating the usage of delta NZCE as defined below:

$$\delta\, P(NZCE) = |\, P(ZCE_{+\,+}) \;-\; P(ZCE_{-\,-})\,| \tag{5.10}$$

Using the feature defined in equation 5.10 might introduce one more feature that can be obtained by using the time series events for the discrimination between speech and music. In

order to examine the performance of the δ P(NZCE) feature, 3000 speech tracks and 3000 music tracks of 3 seconds duration each were examined; see listings A16 and A17.

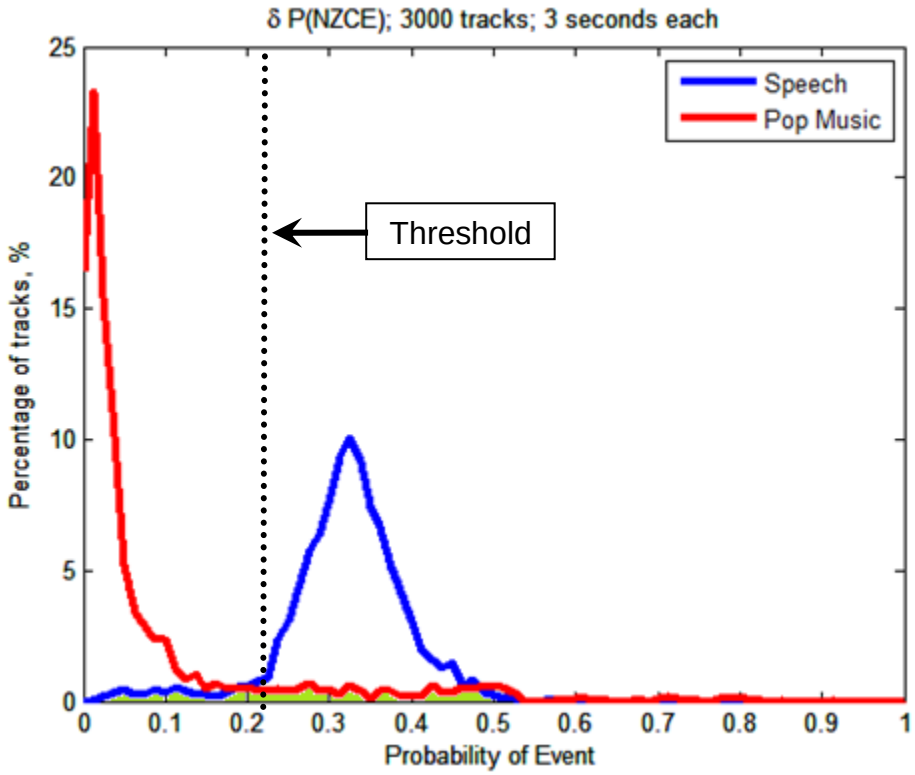


Figure 5.15 The distributions of δ P(NZCE) for speech and music

Figure 5.15 illustrates the distributions that were obtained when using δ P(NZCE) as a feature for both speech and music audio tracks. It is evident from these distributions that there is slight overlap between the speech and the music distributions. To assess the amount of correct and false detection of either speech or music, ROC curves are utilized again as illustrated in figure 5.16. Optimum speech and music detection can be achieved for thresholds of 0.23 and 0.24, respectively. This results in 93.8% of correct speech detection while 12.3% of music tracks being misclassified as speech. Optimum music detection, on the other hand, provides about 88% correct classification at the cost of 6% misclassification.

While this feature performs far better than any ZCE feature, the NZCE features are still the best features in the space of events.In conclusion, using individual NZCE features, i.e. $P(NZCE_{++})$ and $P(NZCE_{--})$, is better than the δ P(NZCE) feature. The latter is better than the ZCE features.

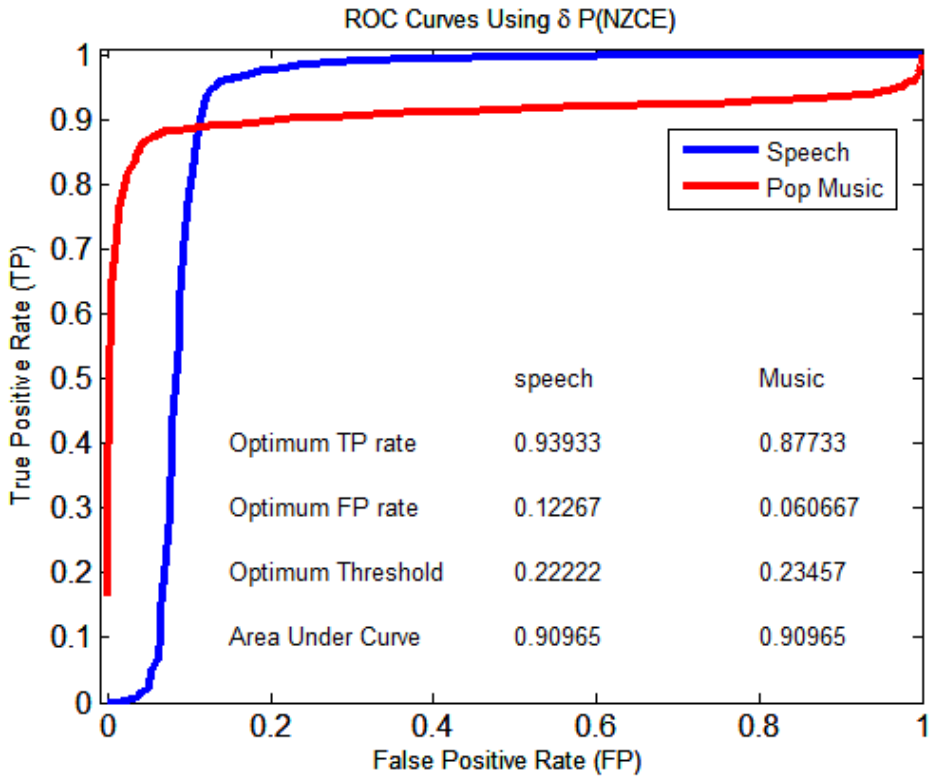


Figure 5.16 ROC curve for using δ P(NZCE) to detect speech audio tracks

5.9. Classification by clustering

Consider the case where an audio signal is identified by two features at the same time instead of a single feature. Using a pair of features in the form of (f_1, f_2) to describe a signal would lead to a different way of classification. So far, several individual features have been thoroughly examined to assess their performance in discriminating between speech and music.

The following fundamental features were examined individually:

1. ZCE features:
    a. $P(ZCE_{+-})$
    b. $P(ZCE_{-+})$
2. NZCE features:
    a. $P(NZCE_{++})$
    b. $P(NZCE_{--})$

Any two features could be used to form the desired pair (f_1, f_2). The following figures illustrate what happens when $f_2$is plotted against f_1 for every pair of features. For every track of audio two parameters are computed. The first parameter against the second parameter would represent a point on the graph.

As expected from the previous analysis of the four features there should be some overlap between the speech and music features. This would appear in figures 5.17 to 5.19 in the form of red dots interfering with the blue cluster of dots. A linear discriminator can be represented by a straight line whose aim is to distinguish between speech and music pairs. To some acceptable level of discrimination, two linearly separable groups can be observed in figures 5.17 and 5.18.

Each pair of features results in two distinctly clustered regions of speech and music. Finding the optimum classifier is not the aim of this thesis. However, it is important to point out that the previous figures illustrate that speech and music can be classified using a simple perceptron Neural Network (NN) to form two linearly separable regions. Furthermore, a clustering NN could be used to classify audio signals into speech and music as demonstrated in figure 5.19.

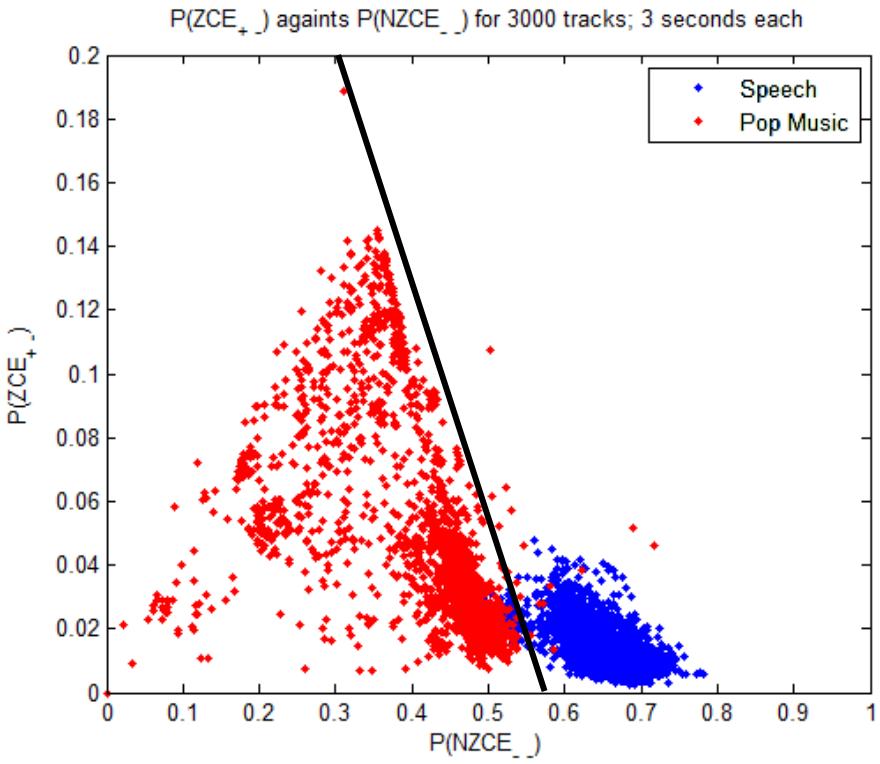


Figure 5.17 A pair of features showing two linearly separable clusters

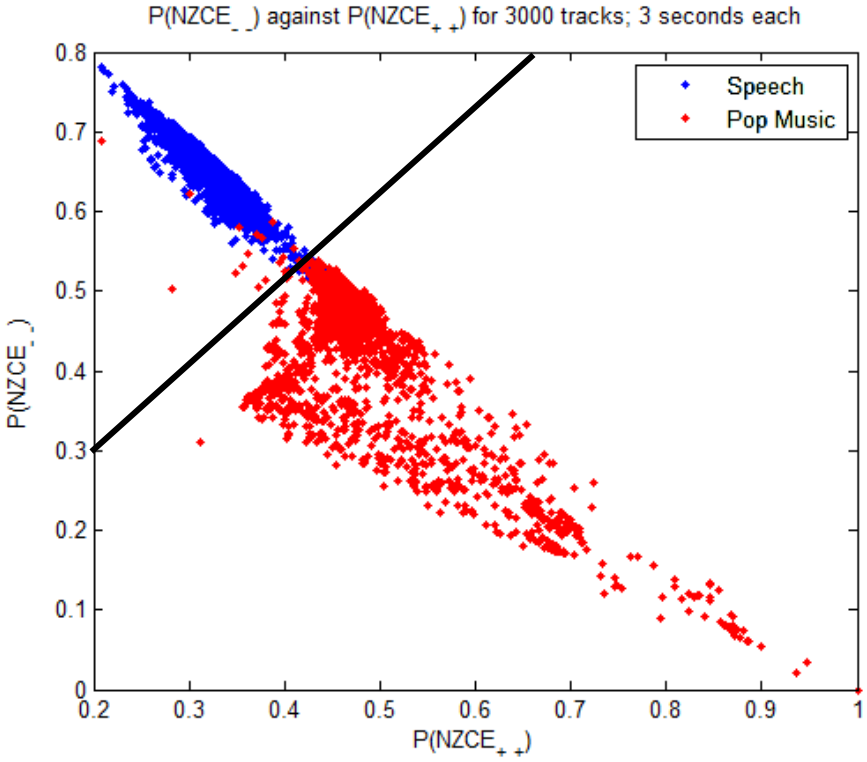


Figure 5.18 A pair of features showing two linearly separable clusters

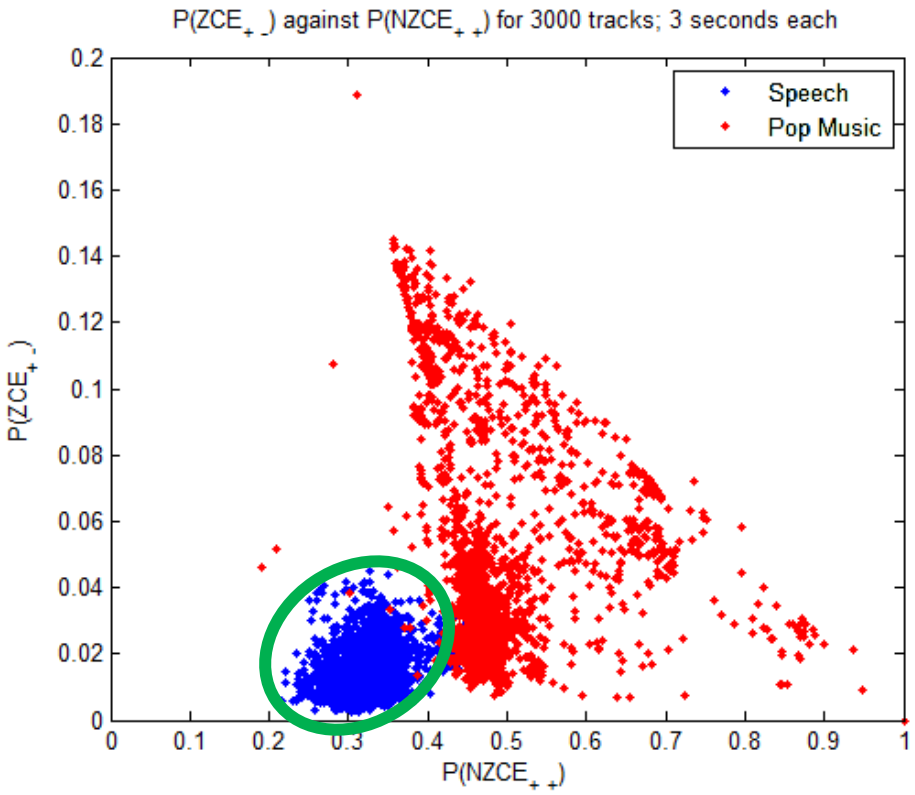


Figure 5.19 A pair of features showing two linearly separable clusters

5.10. Summary

The feature presented in chapter 4 was at the frame level. Its outcome was submitted to the IET Journal as an electronics letter and is under review. On the other hand, this chapter presents other features at the track level. The outcome of the features proposed in this chapter in analysed in both chapters 5 and 6 was presented in two IEEE conferences and can be obtained from ***IEEE Xplore*** website as shown in figure B.1 in Appendix B.

Chapter 5 introduced a couple of novel audio features based on the analysis of the possible time series events (TSE) that could occur in a digital audio signal. It identified a space of events that is composed of 9 possible events and examined the probability of each event to occur in speech signals and in music signals. It was found that two types of events had the

highest probability of occurrence as well as the highest discrepancy between speech and music. They are the event of two successive positive samples, i.e. P($NZCE_{++}$), and the event of two successive negative samples, i.e. P($NZCE_{--}$). The two types of events formed one set in the space of events; i.e. P(NZCE) set. The other set that was examined was the event of a zero crossing which existed in two forms, P(ZCE_{+-}) and P(ZCE_{-+}), which both had low probability of occurrence as well as low discrepancy between speech and music. ROC curves were used to compare the four types of events and their two sets. Based on that assessment they can be sorted from best to worst as follows:

1. P($NZCE_{--}$)➔ Best
2. P($NZCE_{++}$)➔Excellent
3. P(ZCE_{+-}), P(ZCE_{-+}), P(NZCE) set, P(ZCE) set➔Usable
4. Other events: P(Event $_{00}$), P(Event $_{0+}$), P(Event $_{0-}$), P(Event $_{+0}$), P(Event $_{-0}$) ➔ Worst

In this chapter, the possibility of speech/music discrimination by means of clustering of a couple of features was also demonstrated.

The following chapter will analyse the time series events (TSE) even further. It will investigate the effect of the track length, noise, music genre, and sampling rate on the performance of the TSE features. Chapter 6 will also examine individual music instruments as well as various music genres.

Chapter 6: Further Analysis of the Time Series Events

6.1. Introduction

Chapter 5introduced the concept of the time series events (TSE) and the probability of occurrence for each type of event. After thorough experimentation and analysis of all features it was concluded that two of the features performed the best: the $P(NZCE_{+\ +})$ and the $P(NZCE_{-\ -})$.

This chapter will look into various parameters in order to investigate their significance in the context of speech/music discrimination. The ROC curves are again adopted in order to illustrate any change in performance when analysing any factor.The closer the curve is to the ideal curve the better the discriminator is. This implies a higher True Positive (TP) rate and lower False Positive (FP) rate.

The following parameters and cases will be analysed in the subsequent sections of this chapter:

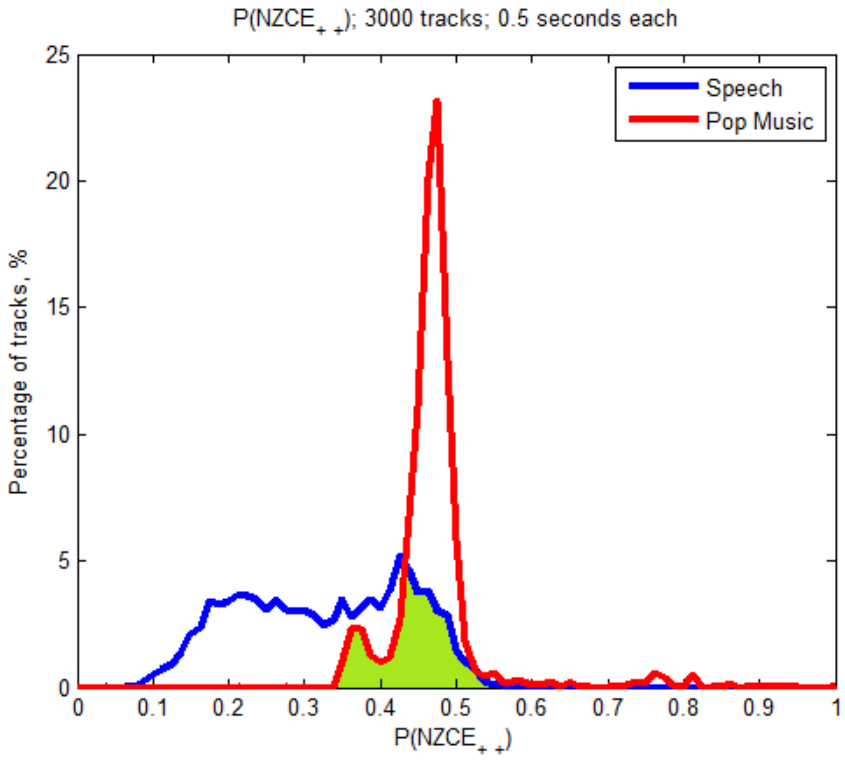
- the effect of using longer tracks
- the effect of adding noise to the clean audio
- the effect of using individual music instruments over a genre of music
- the effect of testing various music genres
- the significance of varying the sampling rate

6.2. Track Length

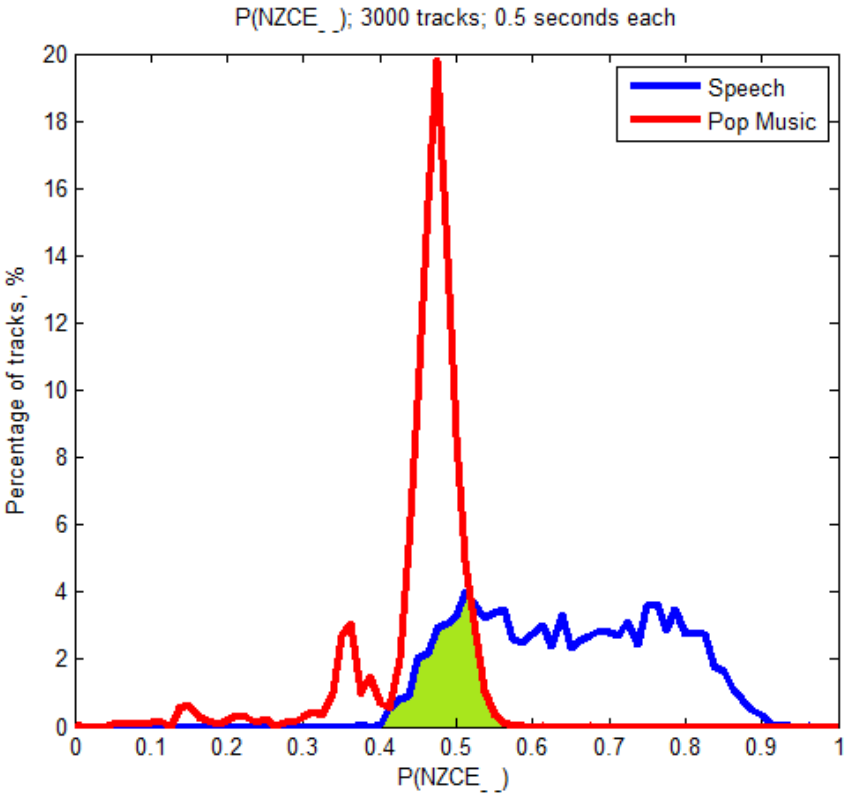
Real time applications consider time delays to be crucial in their design. Although the word "real" is application-dependent, system designers try to minimize the delay of the processing system while maintaining an acceptable performance. This leads to a trade-off between performance and processing delay. The latter is dependent on the complexity of the algorithm, the circuitry of the hardware used, and the length of the processed data.

In a speech/music discrimination system the track length of an audio signal is a predominant factor. In this section, the effect of varying the track length is investigated. Starting with a track length of 0.25 of a second, the track length was gradually increased in steps of 0.25 of a second until a track of 10 seconds was reached. The code scripted to carry out this analysis is shown in listing A18. While the shortest track (0.25 of a second) is expected to result in a big overlap between the speech and music distributions, the longest track length (10 seconds) is expected to have far less overlap. While the track length is increased, the number of analysed tracks was kept constant at 3000 tracks. In other words, 3000 speech tracks and 3000 music tracks were analysed and used to obtain the distributions that were used to produce the distributions in the following figures.

The figures on the following page show the amount of overlap between speech and music distributions for various track lengths. It can be seen from figures 6.1 to 6.4 that increasing the track length has a significant effect on reducing the overlap between the speech and music distributions. This relates back to the probability theory. Computing the probability of an event becomes more accurate when the number of events is increased; i.e. higher population. Thus, increasing the track length is expected to provide better performance.

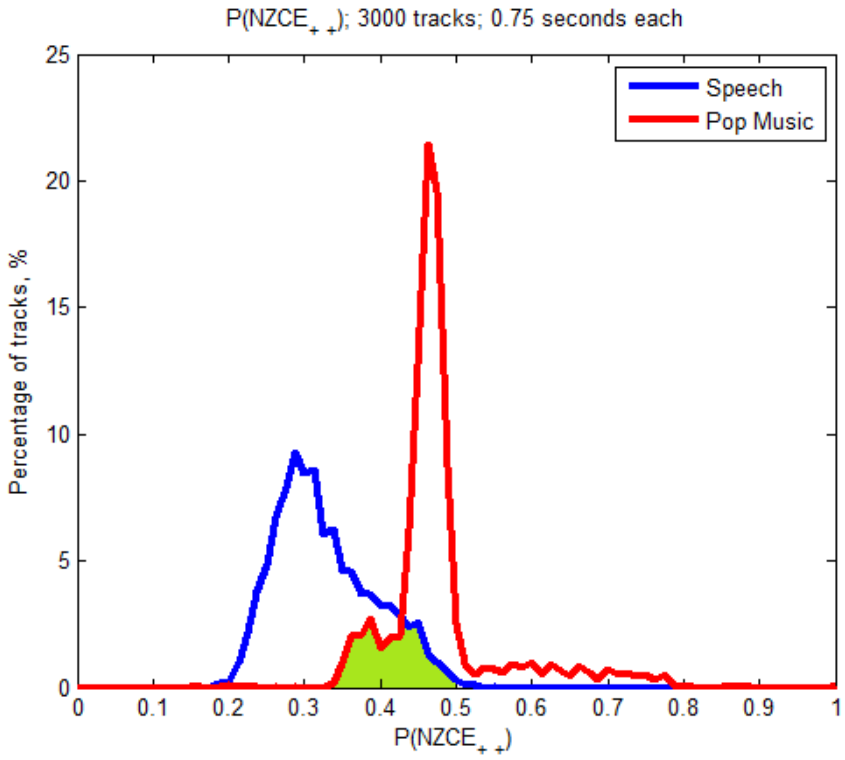


(a)

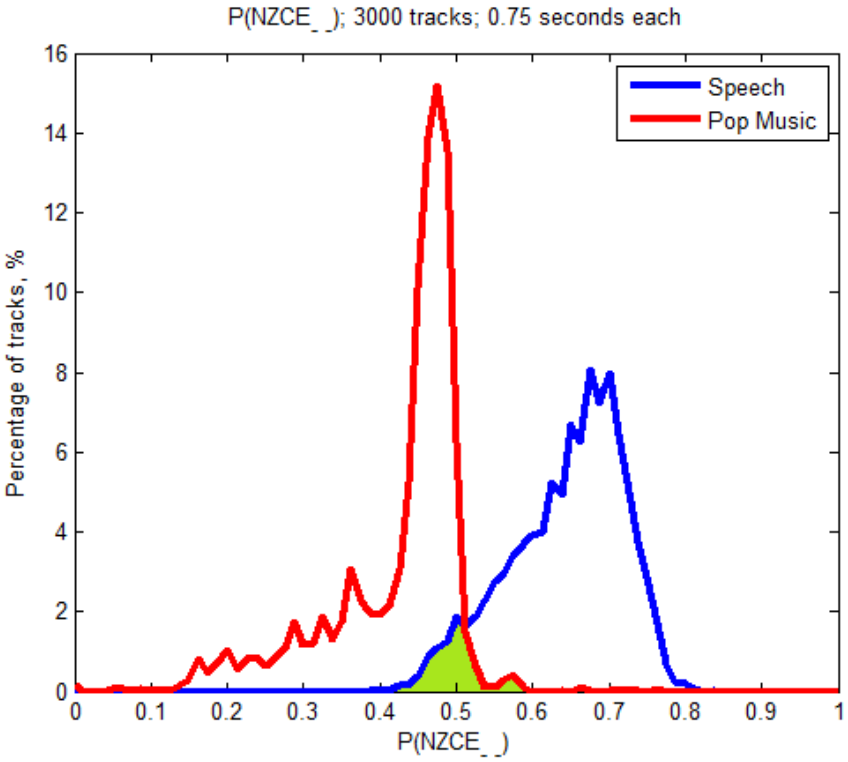


(b)

Figure 6.1 Distribution of $P(NZCE_{++})$and $P(NZCE_{--})$for 3000 speech/music tracks; each 0.25 seconds long

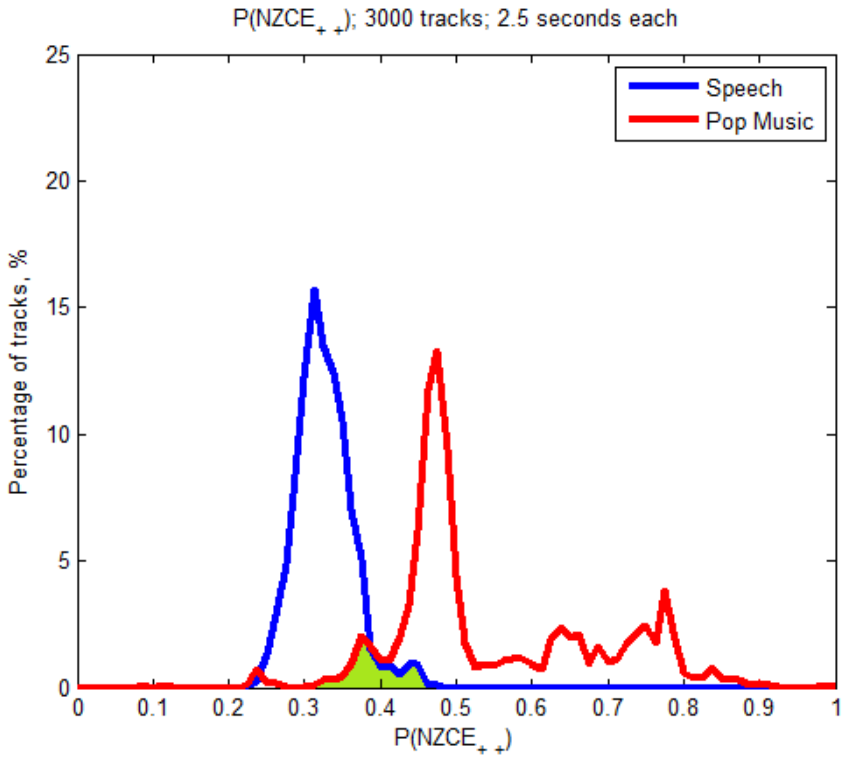


(a)

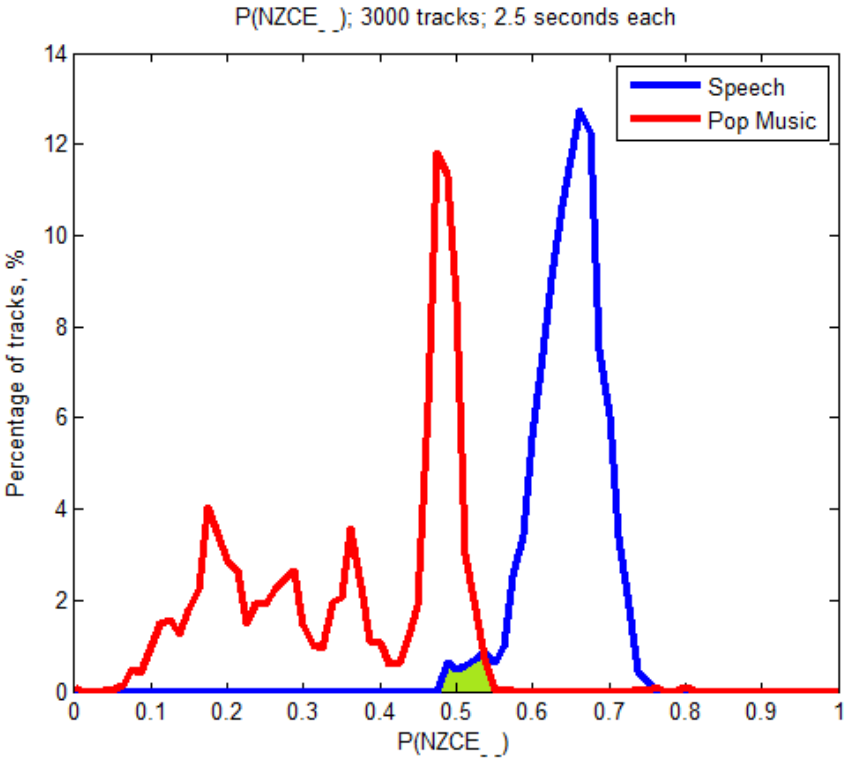


(b)

Figure 6.2 Distribution of $P(NZCE_{++})$ and $P(NZCE_{--})$ for 3000 speech/music tracks; each 0.75 seconds long

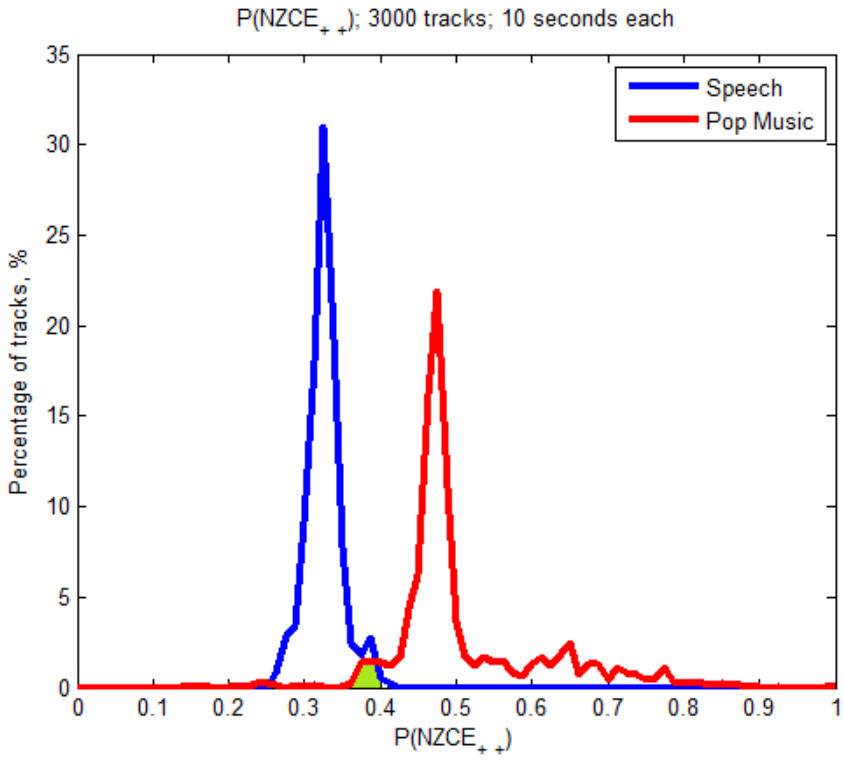


(a)

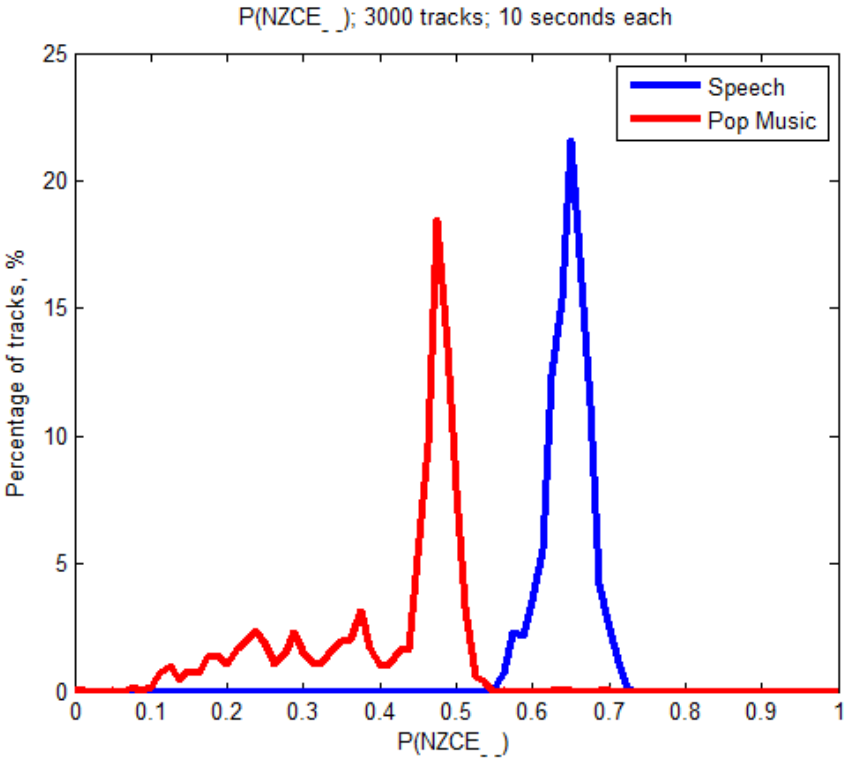


(b)

Figure 6.3 Distribution of $P(NZCE_{++})$ and $P(NZCE_{--})$for 3000 speech/music tracks; each 2.5 seconds long



(a)



(b)

Figure 6.4 Distribution of $P(NZCE_{++})$ and $P(NZCE_{--})$for 3000 speech/music tracks; each 10 seconds long

Throughout all cases in figures 6.1 to 6.4 it can be observed that the $P(NZCE_{--})$ is better than the $P(NZCE_{++})$. Furthermore, the distributions can be divided into 3 regions: definitely speech, definitely music, and fuzzy region; i.e. overlap shaded area. The overlap is reduced to almost zero for the case of 10 seconds for the $P(NZCE_{--})$ shown in figure 6.4.

In order to appreciate how critical the choice of track length can be, consider for example the case of a radio receiver that scans for news channels while skipping music channels. The designer wants to guarantee the detection of the correct type of channel spending adequate time to count the number of NonZero Crossing Events. Thus, it is required to have a buffer of sufficient size to hold a sample recorded from the analysed radio channel. Once a definite classification is made about the type of channel, the radio receiver would either store the frequency of that channel, if it was broadcasting speech, or skip to the next channel if it was broadcasting music. However, if the outcome of the analysis is in the fuzzy region, the system would buffer another track and analyse it in an attempt to make more certain decisions about the analysed radio channel. The designer of such a system can choose between the following options:

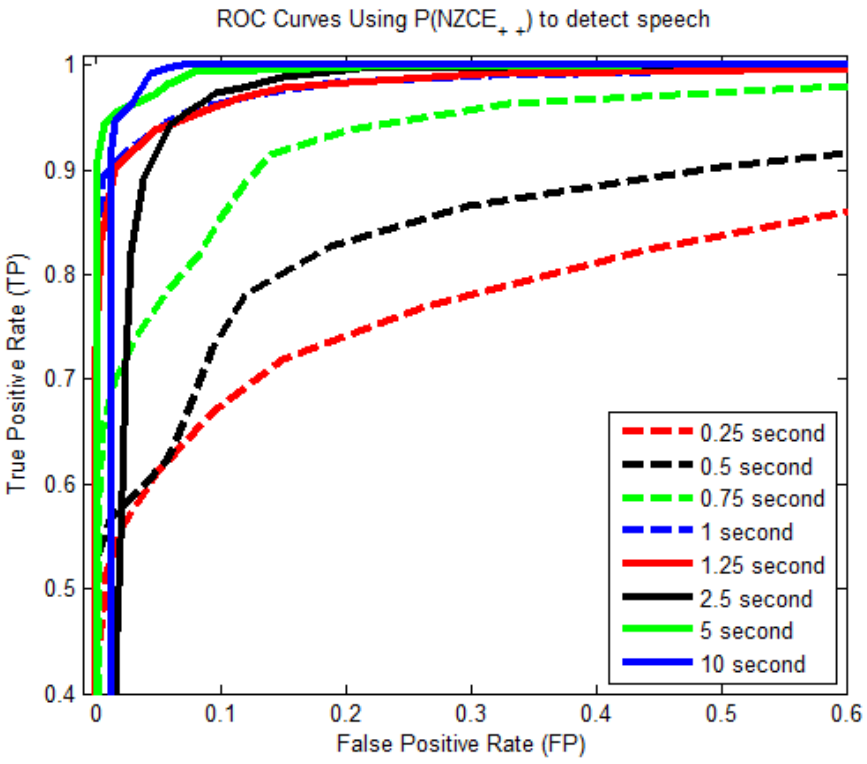
1. A big buffer (e.g. 10 seconds) with higher certainty of type detection and very small fuzzy region, leading to waiting longer to get an output
2. A small buffer (e.g. 2 seconds) with lower certainty of type detection and relatively bigger fuzzy region

If the second choice was selected, the system might be designed in such away as to iterate the detection process several times before it is able to make a certain decision about the analysed channel. However, after 5 iterations, both options would cost roughly an equal amount of time.

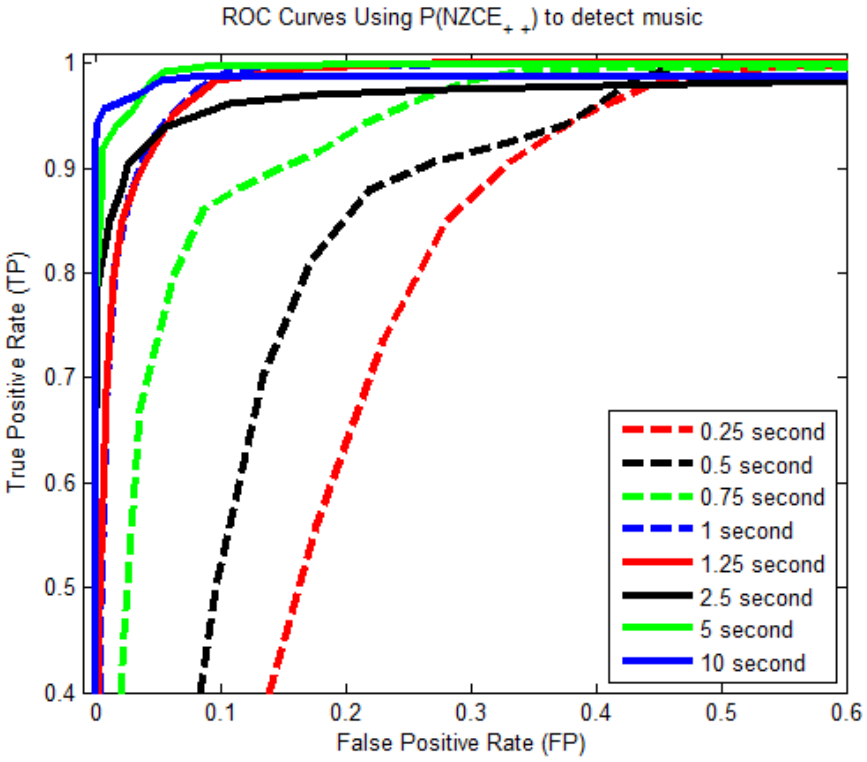
Furthermore, ROC curves can be used to measure the performance of the $P(NZCE_{--})$ and $P(NZCE_{++})$ as the track length is increased it becomes more obvious as shown in figure 6.5. Changing the track length from 0.25 of a second to 1 second shows significant enhancement in detecting speech. However, there is insignificant enhancement when comparing 1 second to 5 seconds. For real-time applications, it might not be worth it the delay for such little enhancement. In data retrieval systems any slight enhancement is of a significant value since there is no restriction on the track length.

The influence of increasing the track length can also be demonstrated by observing the optimum point on each ROC curve. Figure 5.6 illustrates the performance obtained when using various track lengths for both features: $P(NZCE_{++})$ and $P(NZCE_{--})$. For the former one, when detecting speech at 0.25 seconds track length, 0.72 True Positive (TP) rate was obtained at the cost of 0.15 False Positive (FP) rate. However, when detecting music at the same track length, 0.85 TP rate was obtained at the cost of 0.28 FP rate. Thus, there is certain discrepancy between speech and music detection performance at very low track length.

The discrepancy between speech detection and music detection performance vanishes as the track length was increased. At about 0.75 seconds track length, speech detection and music detection TP/FP rates almost coincide with each other. Also, it can be noticed that there is no significant enhancement beyond one second track length. Another figure of merit that has been discussed previously is the Euclidian distance from the optimum (TP, FP) coordinate to the ideal point (0,1). As the track length was increased, the Euclidian distance reduced. This is illustrated in figure 6.7. Each graph in figure 6.7 shows two curves for speech and music that coincide with each other; thus only one curve is visible. There is a distinct reduction in Euclidian distance between 0.25 seconds and 1 second. Beyond one second the distance is improved slightly, yet insignificantly.

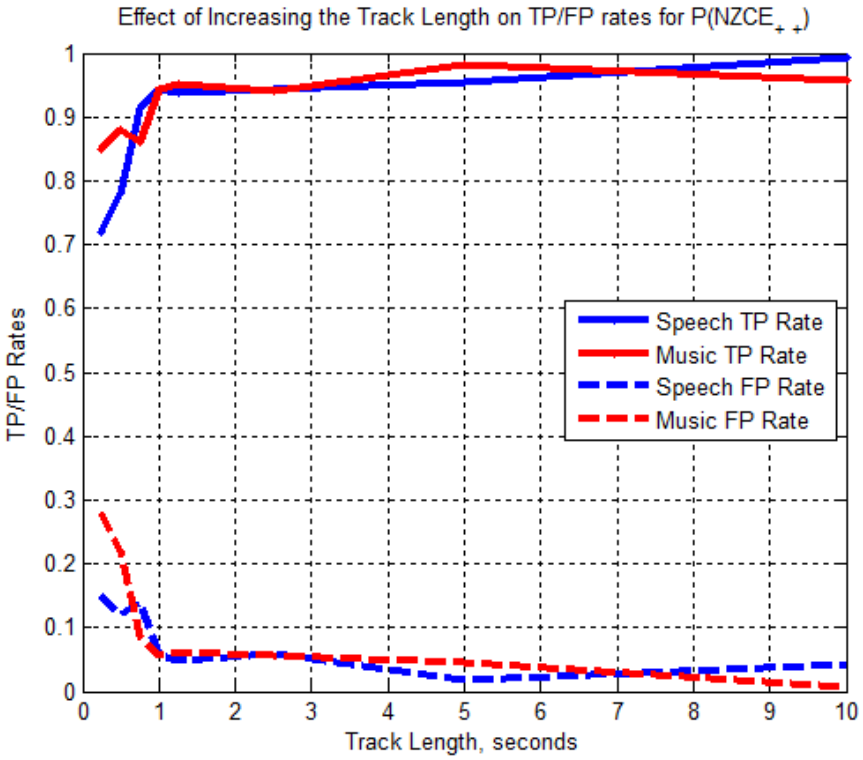


(a)

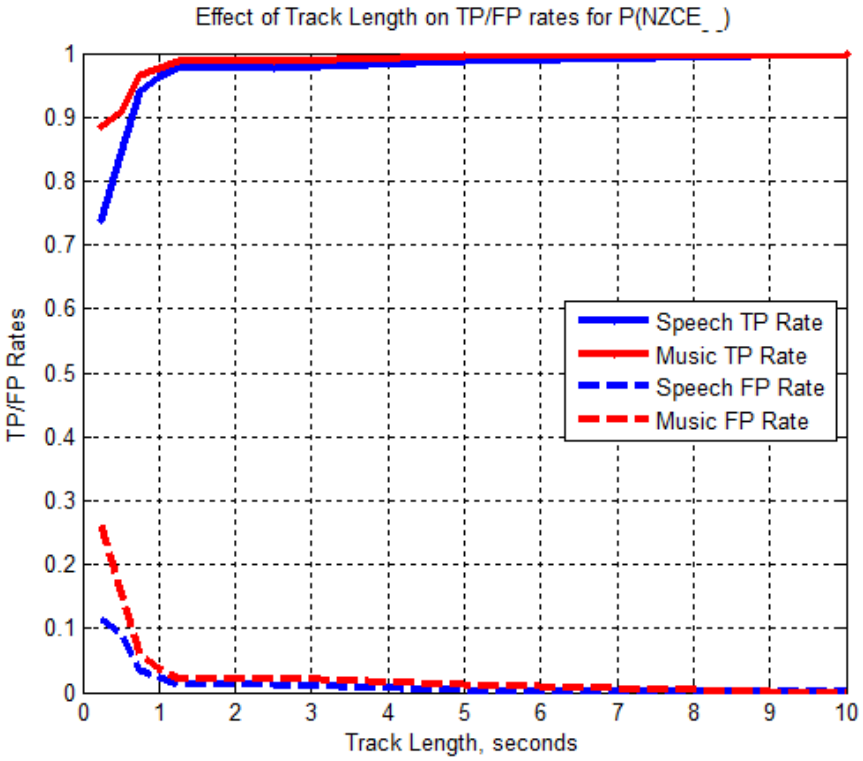


(b)

Figure 6.5 ROC curves enhancement as a result of increasing the track length when detecting speech and music tracks

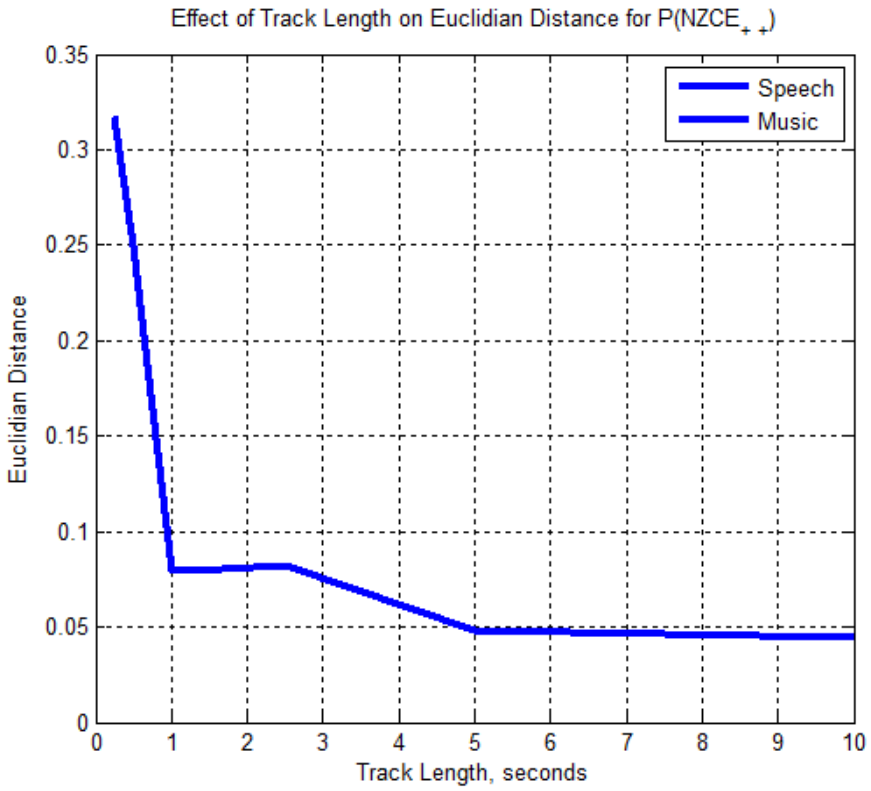


(a)

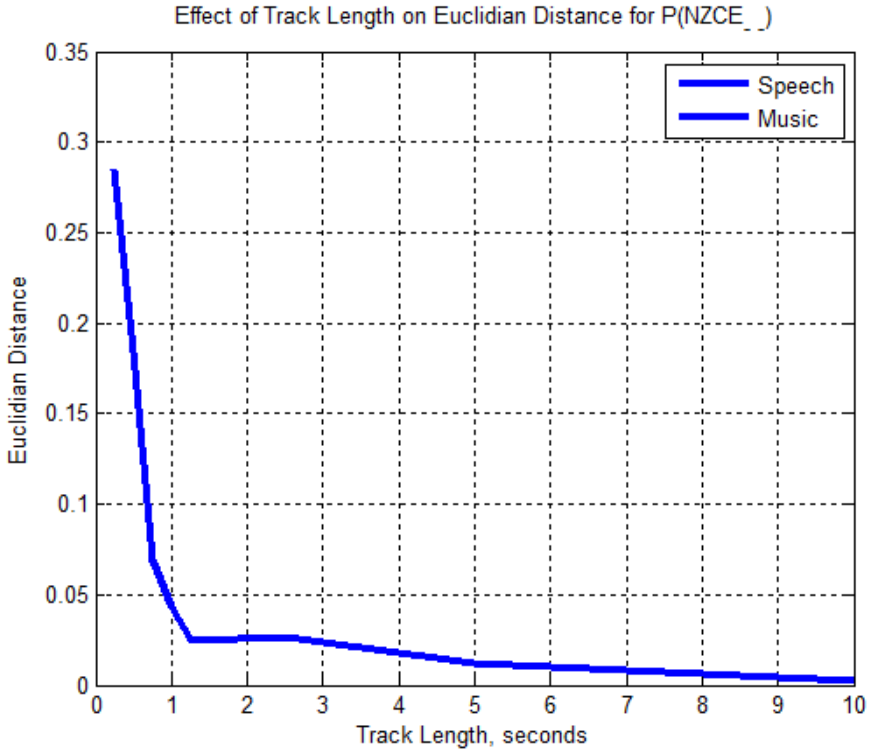


(b)

Figure 6.6 TP and FP rates of as a function of track length for the P(NZCE$_{+\,+}$) and the P(NZCE$_{-\,-}$) feature



(a)



(b)

Figure 6.7 Optimal Euclidian Distance as a function of track length for the P(NZCE$_{+\,+}$) feature and the P(NZCE$_{-\,-}$) feature

6.3. Effect of Noise

In the spectrum of audio signals, noise could take several forms. One type of noise was selected to assess the influence of noise on the proposed features, normally distributed noise is sometimes also called Gaussian noise and has the probability density function (PDF) shown in figure 6.8. The probability density function for a Gaussian distribution can be modelled using equation (6.1):

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (6.1)$$

Where σ is the standard deviation, μ is the mean, and *exp* is the exponential function.

A special case where σ = 1, and μ = 0 is called the standard normal distribution, and with these parameters equation (6.1) can be simplified to the form in equation (6.2) whose distribution is illustrated in figure 6.8.

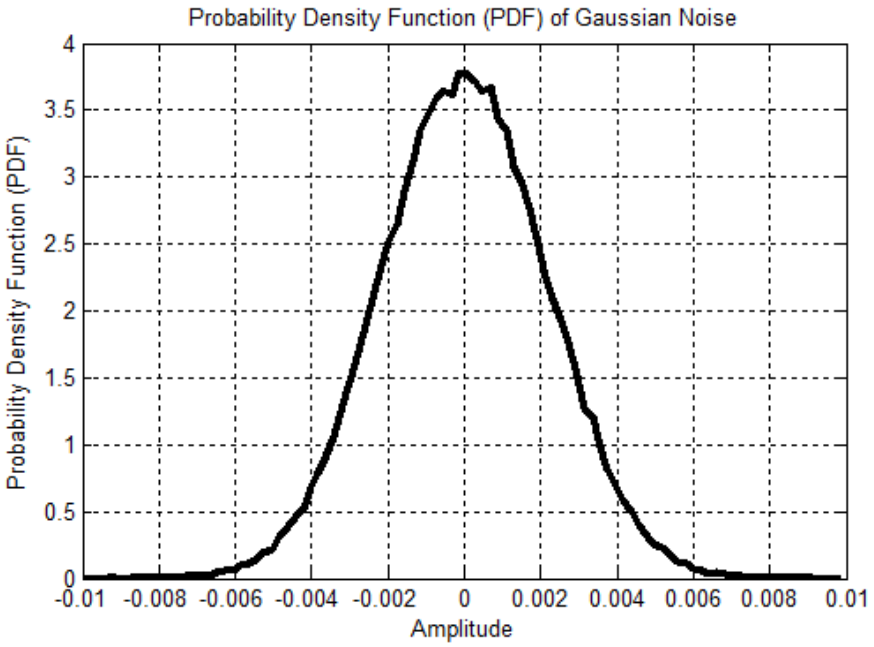


Figure 6.8 The probability density function (PDF) of Gaussian Noise with zero mean

$$p(x) = \frac{1}{\sqrt{\pi}} exp\left(-\frac{1}{2}x^2\right) \tag{6.2}$$

By looking at the spectrum illustrated in figure 6.9 it can be observed that the Gaussian noise is flat throughout the whole spectrum while the speech and music power spectral density is higher at the lower frequencies and is lower at the higher frequencies.

There exist many types of noise which can be related to its spectrum via equation (6.3):

$$S(f) \propto \frac{1}{f^n} \tag{6.3}$$

Where *f* is the frequency and *n* is a factor that can take various values depending on the type of noise.

Table 5.1 shows these types and their corresponding *n* value. The analysis carried out in the following experiments are for the case of n = 0; i.e. white noise. The power spectral density for the white noise is shown in figure 6.9 as compared to that of speech and music signals.

Type of Noise (other names)	**Name by colour**	**n**
Azure noise	Blue noise [57]	-1
Normal (Gaussian)	White noise [58, 59]	0
1/f noise (flicker noise)	Pink noise [60-63]	1
Brownian noise	Red noise [59]	2
Silence[1]	Black noise [57]	> 2

Table 6.1 Types of $1/(f^n)$ noise and their corresponding n values

[1] Noise that has a frequency spectrum of predominantly zero power level over all frequencies (few narrow bands or spikes may be accepted)

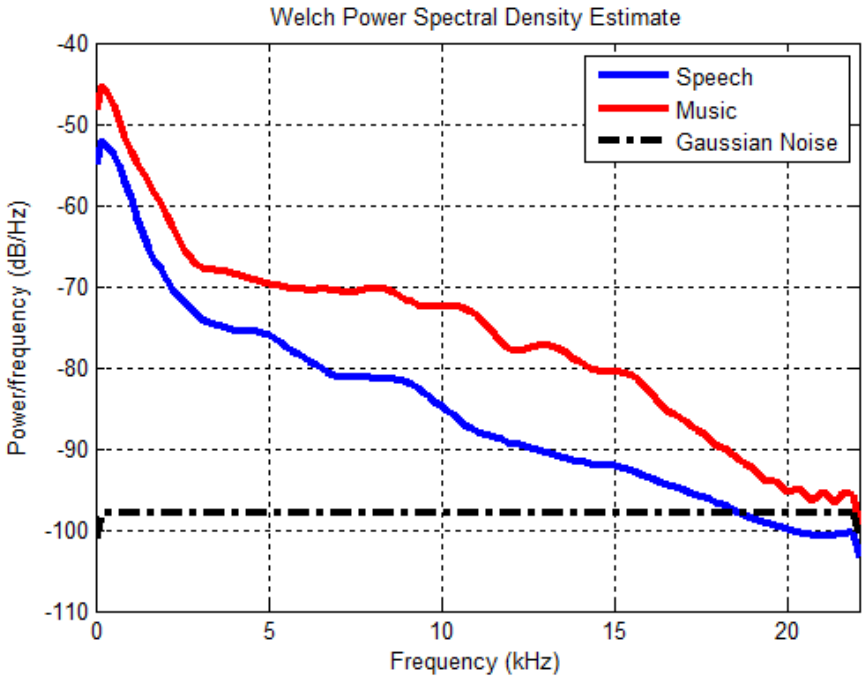


Figure 6.9 The power spectral density of speech, music, and Gaussian noise

Prior to investigating the effect of adding noise to any audio signal on the various time series events, let us look at the white noise alone. How does its distribution look like on the probability axis? By comparing the distributions of speech and popular music to the white Gaussian noise the figures (6.10 to 6.15) can be obtained for various types of events. The analysis is carried out on 3000 speech tracks, 3000 music tracks, and 3000 simulated Gaussian noise tracks. Each track is 3 seconds long and sampled at 44.1 kHz. Refer to listing A19 for the code scripted to carry out the experiments in this section.

By observing figures (6.10 to 6.15) it can be noticed that the three classes of audio, i.e. speech, music, and noise, can be clearly distinguished from each other using any of the two NZCE types (i.e. $NZCE_{++}$ and $NZCE_{--}$). The three types of audio fall into three distinct bands of probabilities. Figures 6.14 and 6.15 show the ZCE set and the NZCE set,

respectively. Although both sets cannot clearly discriminate between speech and music, they can still be used to distinguish between intelligible sounds (i.e. speech and music) and noise.

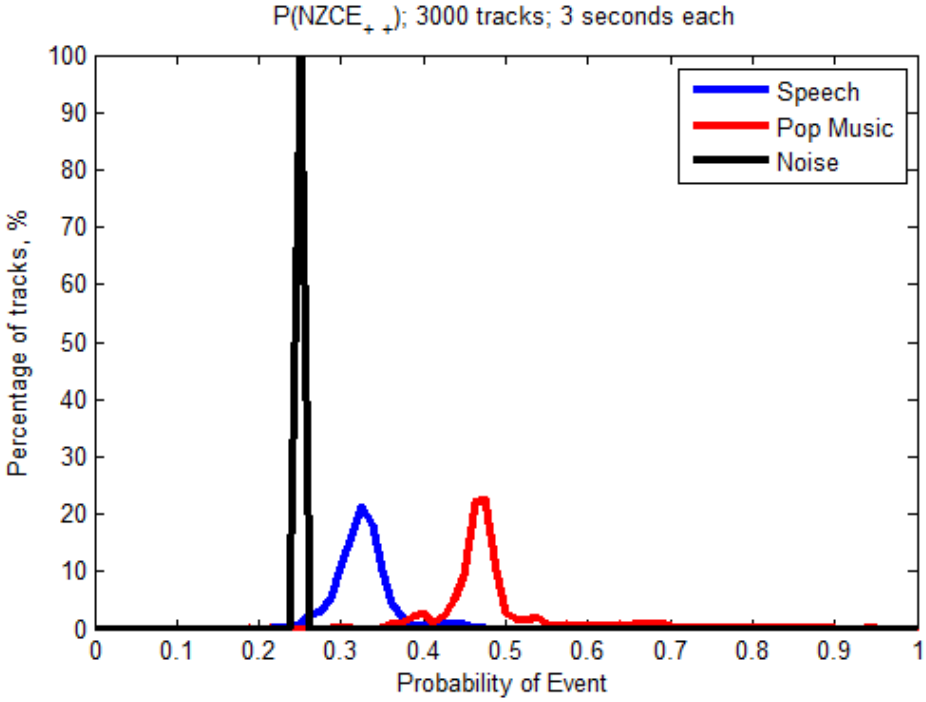


Figure 6.10 The distribution of the P($NZCE_{+\,+}$) feature for speech, music, and noise

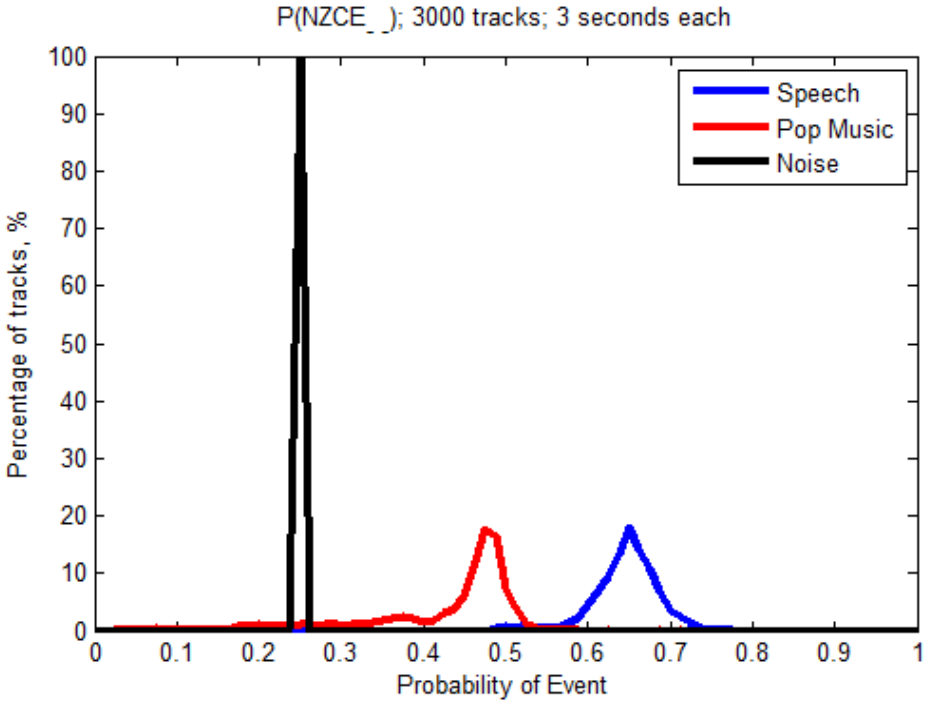


Figure 6.11 The distribution of the P($NZCE_{-\,-}$) feature for speech, music, and noise

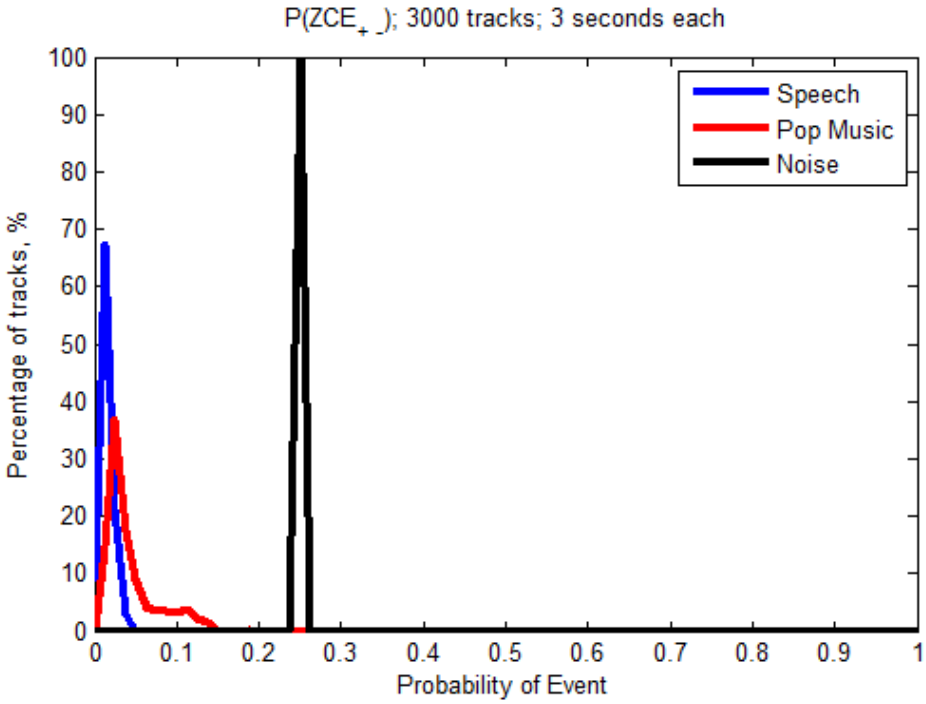


Figure 6.12 The distribution of the P(ZCE$_{+-}$) feature for speech, music, and noise

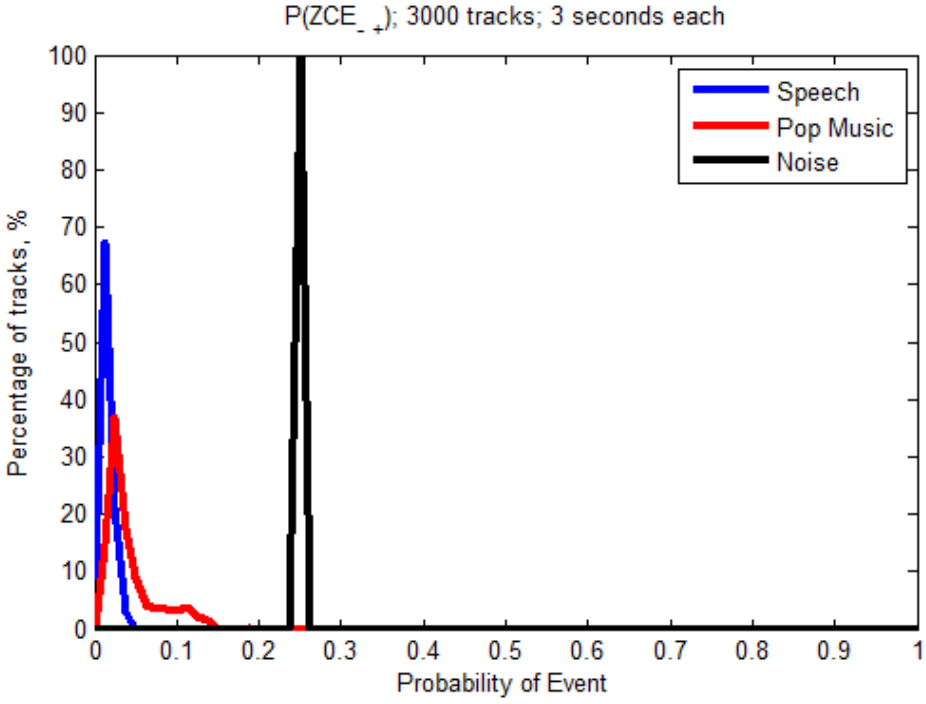


Figure 6.13 The distribution of the of P(ZCE$_{-+}$) feature for speech, music, and noise

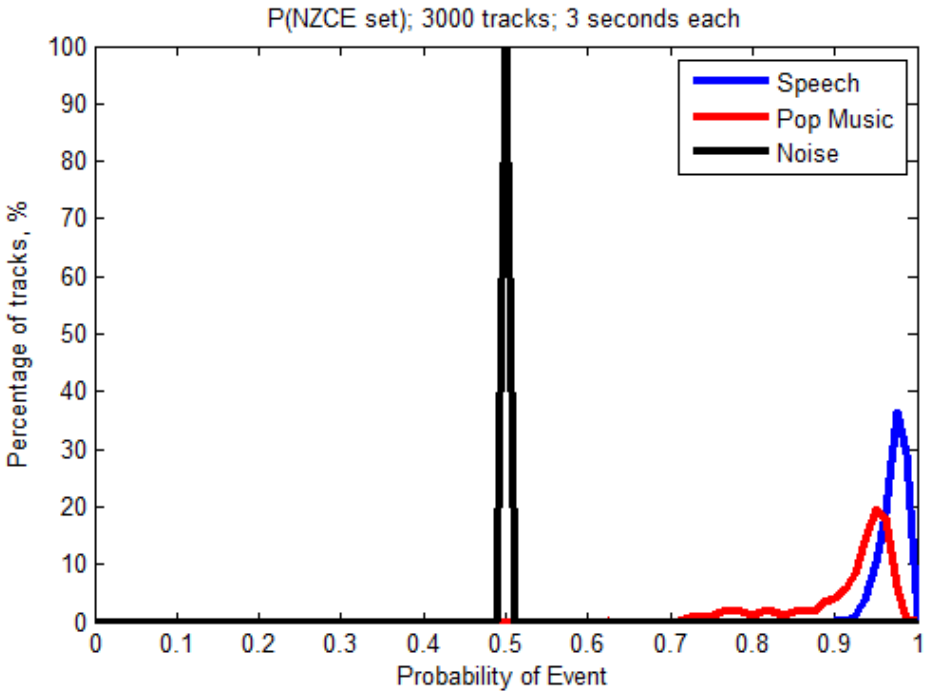


Figure 6.14 The distribution of the P(NZCE set) for speech, music, and noise

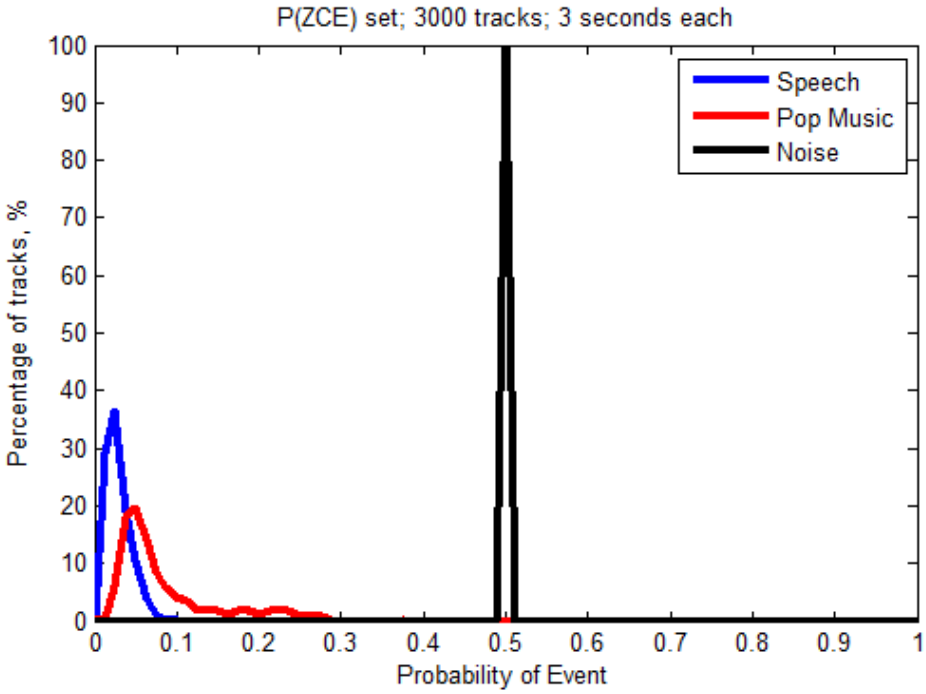


Figure 6.15 The distribution of the P(ZCE set) for speech, music, and noise

Having shown the distribution of the three types of audio signals (speech, music, and noise) let us now examine some noisy speech and music. The audio signals that are used to examine the various audio features throughout this thesis were recorded in an anechoic chamber where noise is very minimal; i.e. close to zero. To be more realistic, 60 seconds of room noise was recorded and linearly added to every 60 seconds of speech and music.

Figure 6.16 and figure 6.17 show5 seconds of the recorded room noise and its probability density function, respectively. The P($NZCE_{++}$) and P($NZCE_{--}$) distributions were then obtained. The Signal to Noise Ratio (SNR) was then computed from:

$$SNR = 20\ log\ \frac{RMS_{speec\ h}}{RMS_{noise}} \tag{6.4}$$

The RMS power can be computed as explained in chapter 3; equation 3.1.

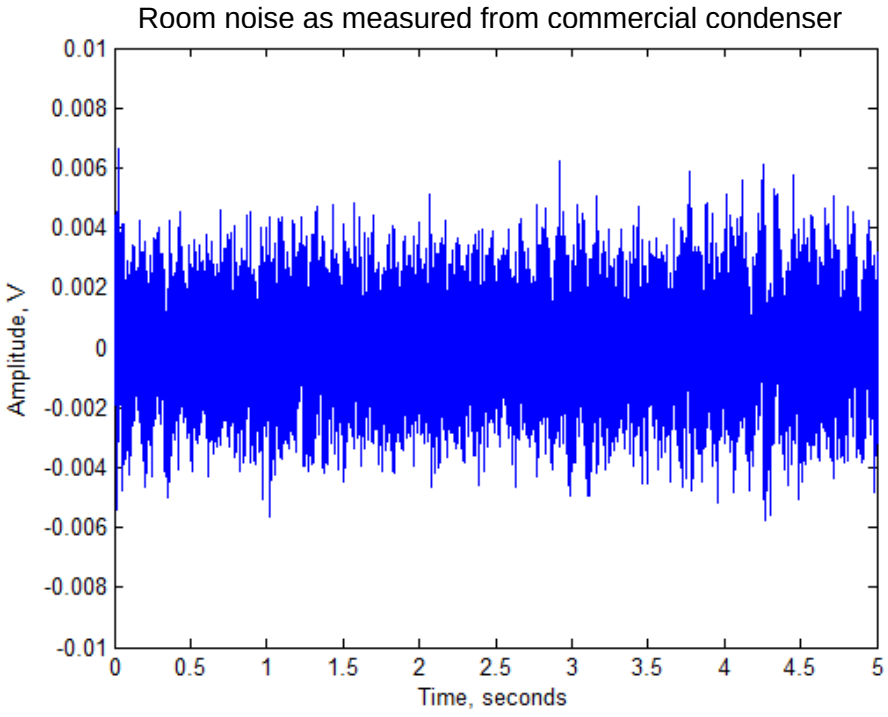


Figure 6.16 Room noise (5 seconds) recorded using a commercial condenser microphone

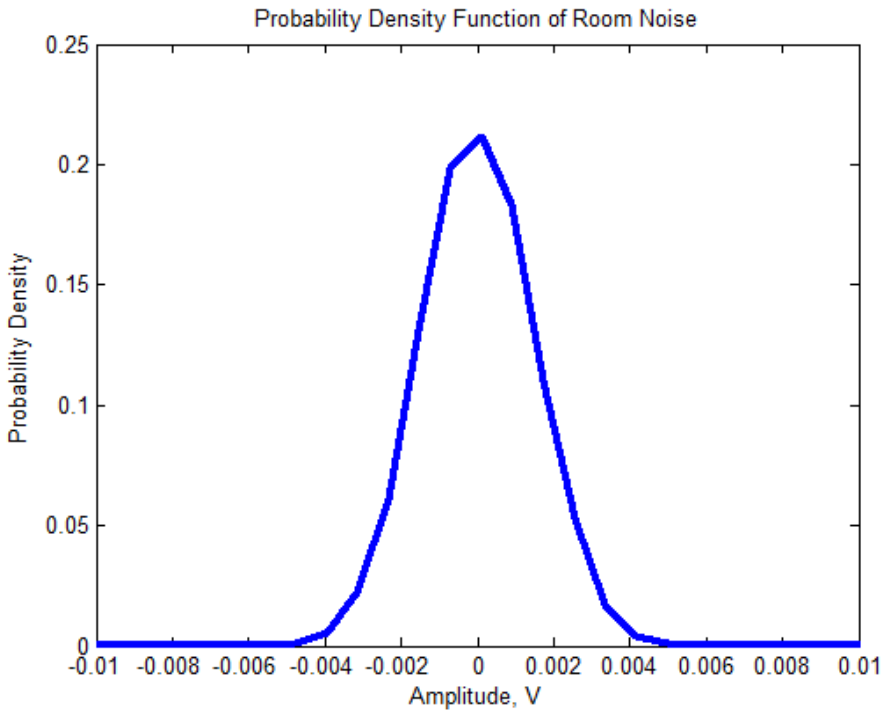


Figure 6.17 Probability Density Function (PDF) of 60 seconds of room noise recorded using a commercial condenser microphone

Figure 6.18 shows the distributions of the $P(NZCE_{++})$ of two levels of SNR. There is certain obvious overlap introduced by reducing the SNR from about 55 dB down to about 33 dB. To examine the effect of increasing the noise power, the recorded noise is amplified by a constant gain. Each time the room noise is amplified further and added to the speech and music, the two features are re-examined to analyse the effect of the SNR. In terms of audibility, as noise becomes more dominant (i.e. noise power increased) the noisy-speech would sound more like pure noise.

Because each feature has a different distribution, the effect of increasing the noise power on the amount of overlap is expected to be different. This is best illustrated by the following figures. The figures (6.19 to 6.22) illustrate, by various means, how sensitive TSE features

are to noise. The higher the noise power is, i.e. lower SNR, the lower the TP rate and the higher the FP rate. In other words, the proposed TSE features are not robust to noise. It would work well in low noise environments.

All means of assessing the implications of noise on speech/music discrimination using the $P(NZCE_{++})$ and $P(NZCE_{--})$ features show how sensitive the two features are to the surrounding noise. In noisy environments the examined audio data could be highly misclassified into the wrong class. However, it is highly unlikely for the recorded audio data to be so low SNR. News broadcast, for instance are recorded in very low noise and RT_{60} ratings. So, a radio channel scanner is expected to have clean speech; i.e. noise level close to the ones provided by the UCL anechoic chamber.

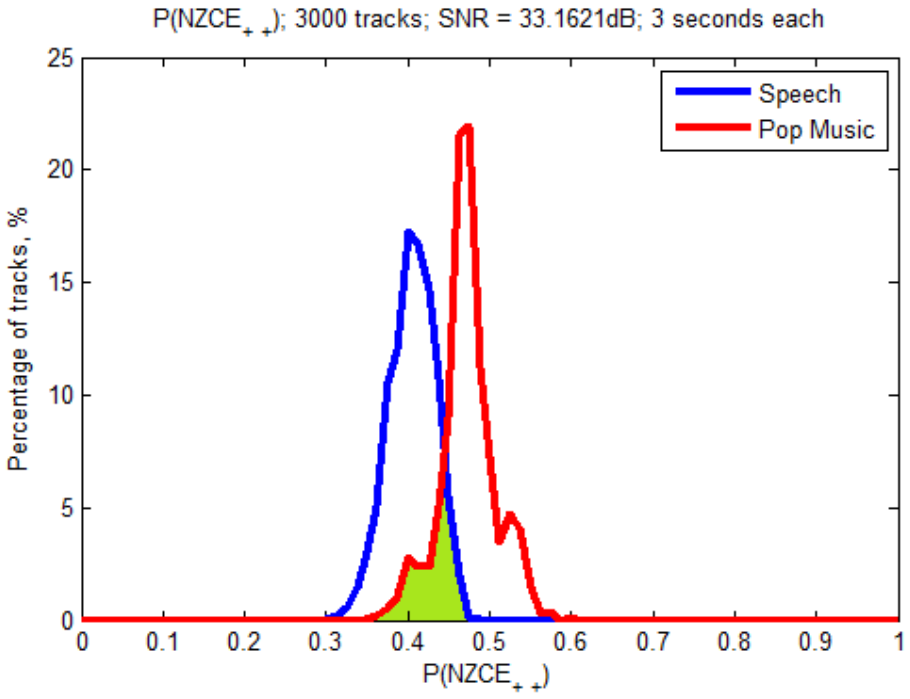
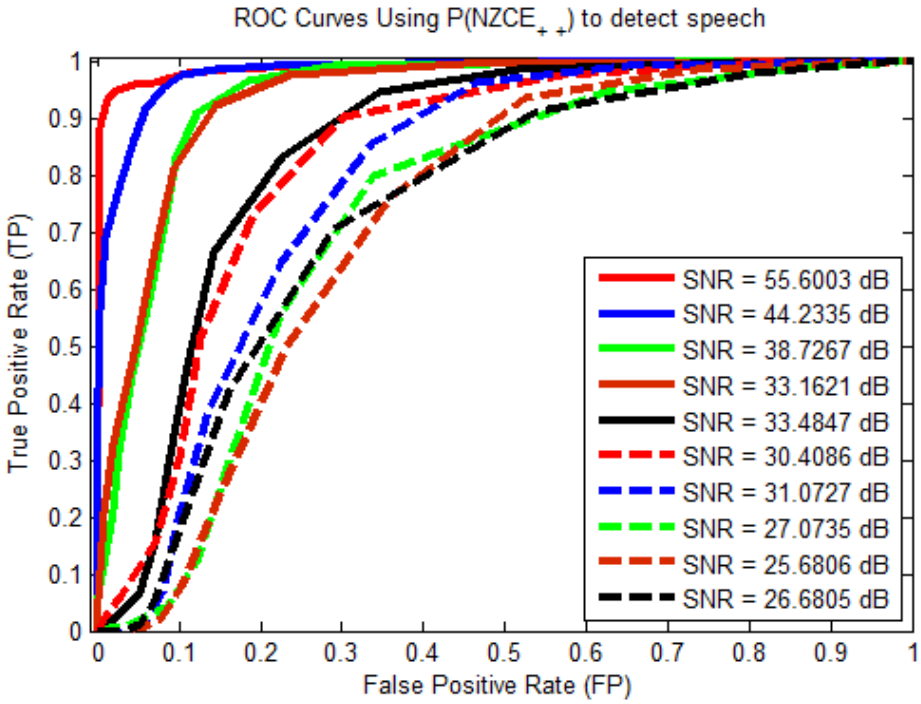
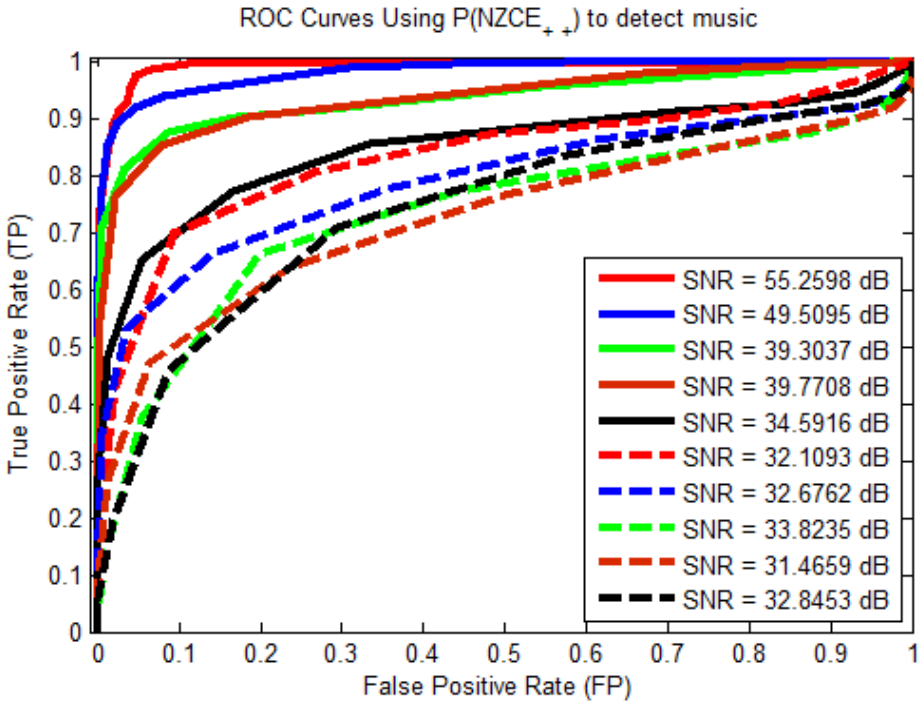


Figure 6.18 The distribution of the $P(NZCE_{++})$ for 3000 noisy speech and music tracks with SNR = 33 dB

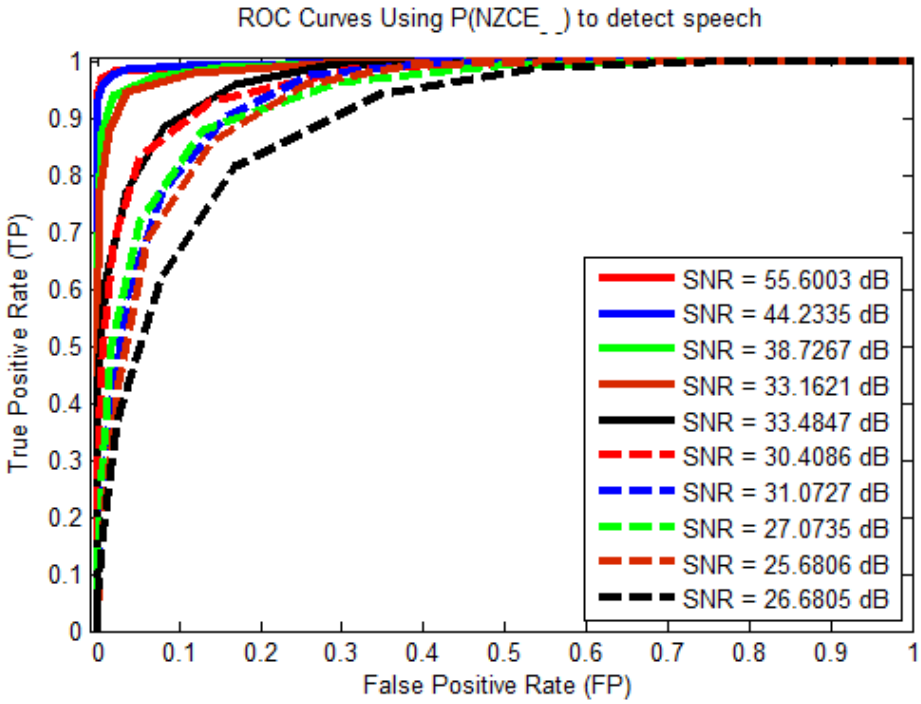


(a)

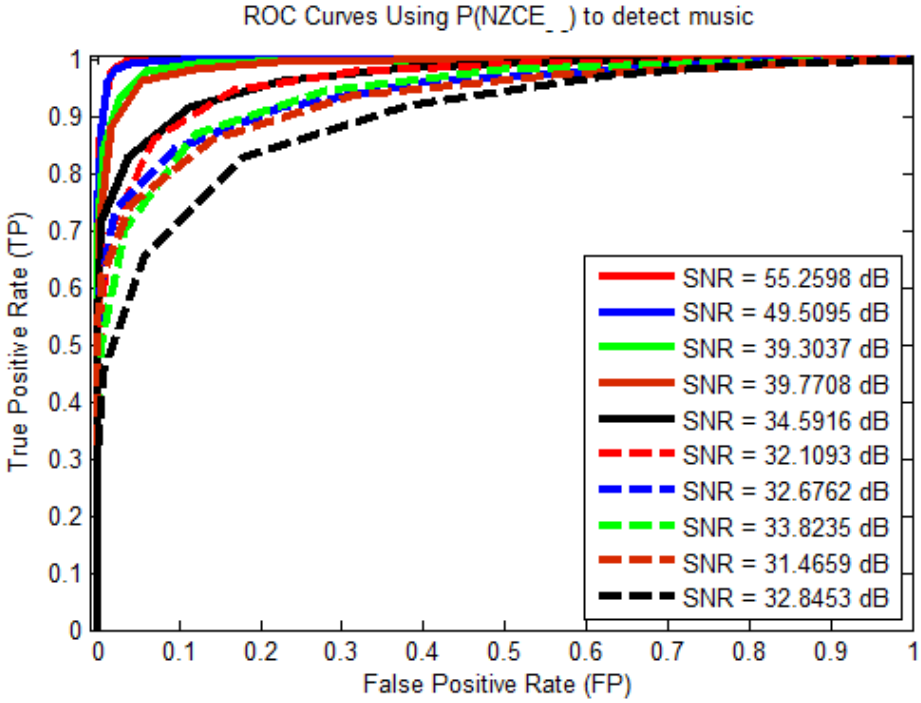


(b)

Figure 6.19 Effect of the SNR on the ROC curves of the P($NZCE_{+\,+}$) for noisy speech and music tracks; 3 seconds each

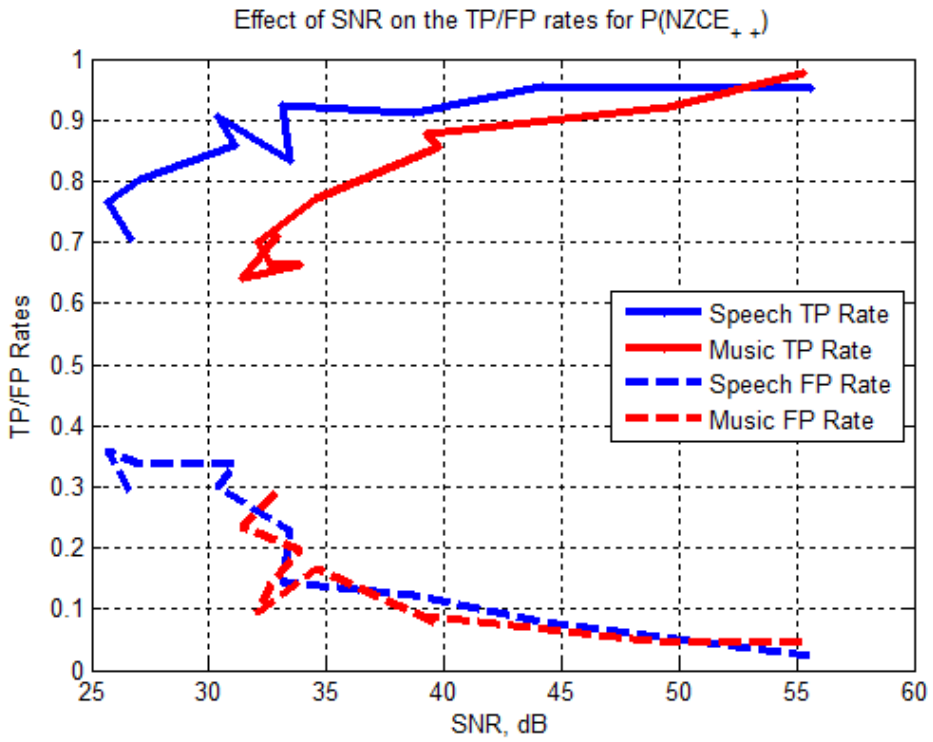


(a)

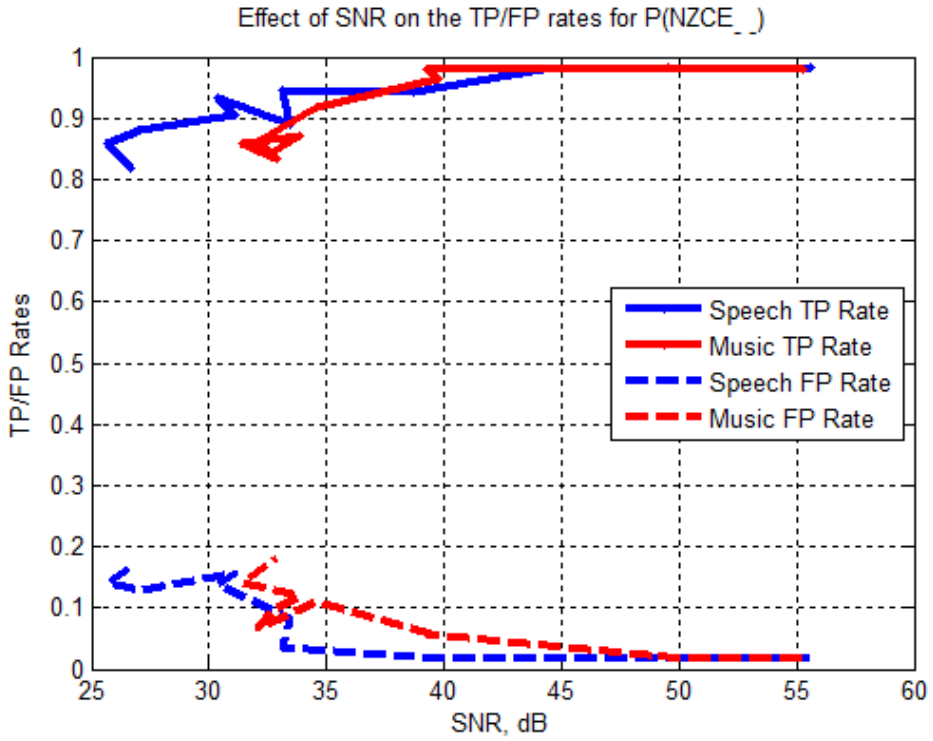


(b)

Figure 6.20 Effect of the SNR on the ROC curves of the P(NZCE_ _) for noisy speech and music tracks; 3 seconds each

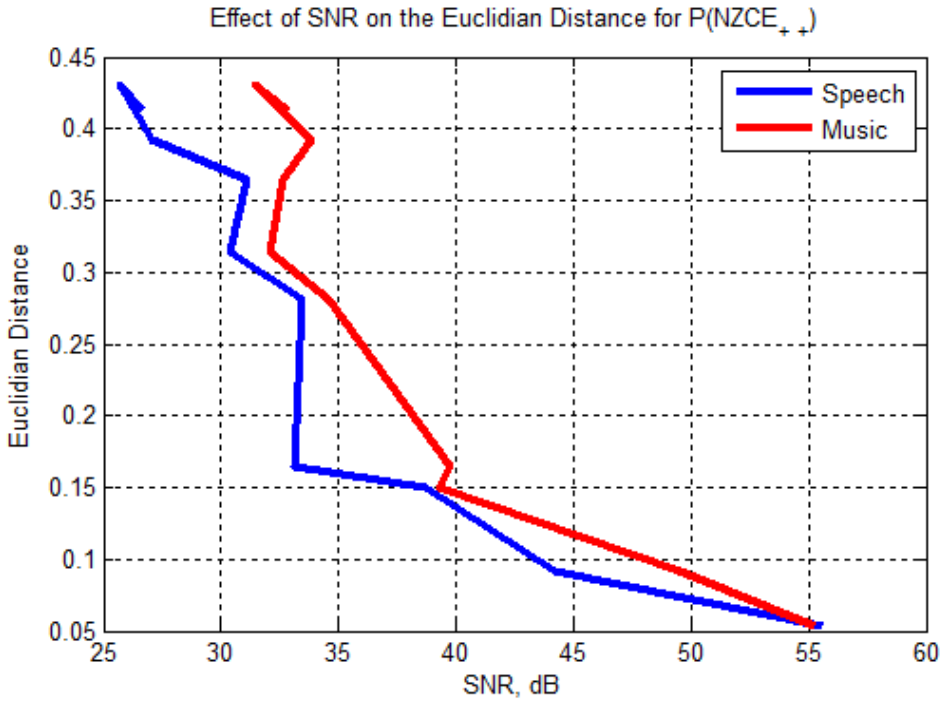


(a)

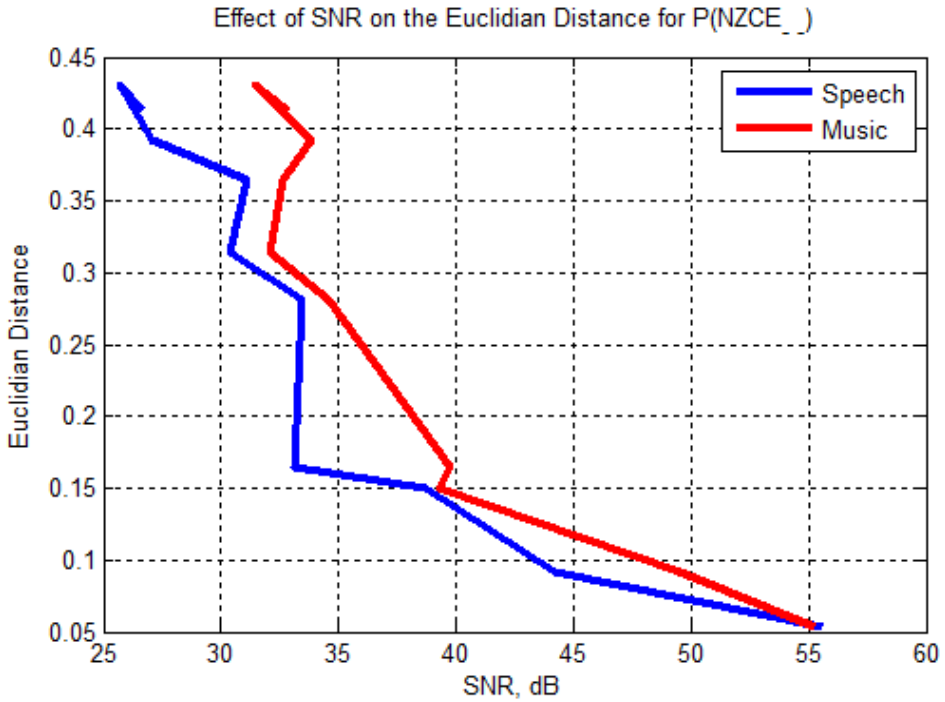


(b)

Figure 6.21 Effect of the SNR on the TP and FP rates of the P(NZCE$_{+\,+}$) and the P(NZCE$_{-\,-}$) for noisy speech and music tracks; 3 seconds each



(a)



(b)

Figure 6.22 Effect of SNR on the P(NZCE$_{+\,+}$) and the P(NZCE$_{-\,-}$) on the Euclidian Distance of the ROC curves of noisy speech and music tracks; 3 seconds each

6.4. Individual music instruments

The analysis carried out so far used popular music for comparison with speech. It is important to consider individual instruments instead of a set of instruments as in normal popular songs.

Let us set a hypothesis that an individual instrument would behave in the same way as a group of instruments. In order to verify this hypothesis, the $P(NZCE_{++})$ and $P(NZCE_{--})$ features were tested for 64 different instruments against the same set of speech. Figures 6.23and6.24 illustrate the $P(NZCE_{++})$ and $P(NZCE_{--})$ of speech tracks as well as 6 different instruments, respectively. Figure 6.25 shows examples of the 6 instruments. Although each instrument shows a different distribution, they share the same band of probability and are very distinguishable from the speech distribution. The ROC curves are not produced here since the similarity between individual instruments and popular music (i.e. a group of instruments) has been proven in principle.

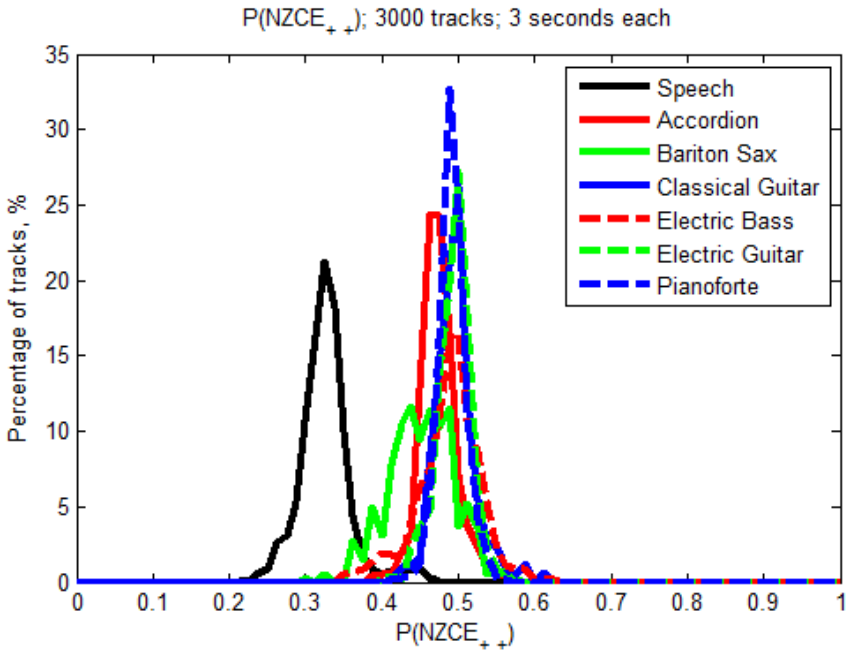


Figure 6.23 Distribution of the $P(NZCE_{++})$ for speech and 6 individual instruments (accordion, Baritone Sax, classic guitar,electric bass, electric guitar, and pianoforte)

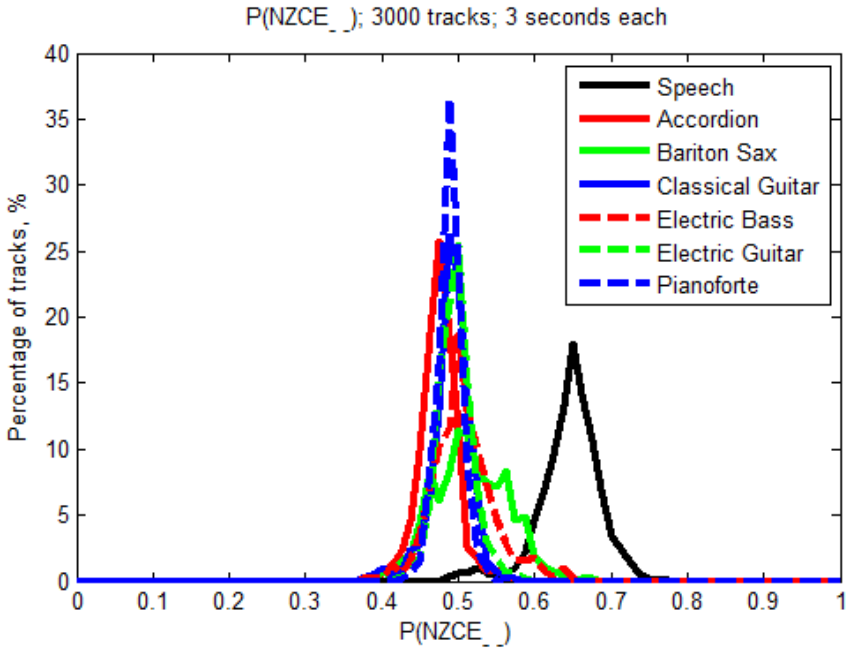


Figure 6.24 Distribution of the P(NZCE_ _) for speech and 6 individual instruments (accordion, Baritone Sax, classic guitar, electric bass, electric guitar, and pianoforte)

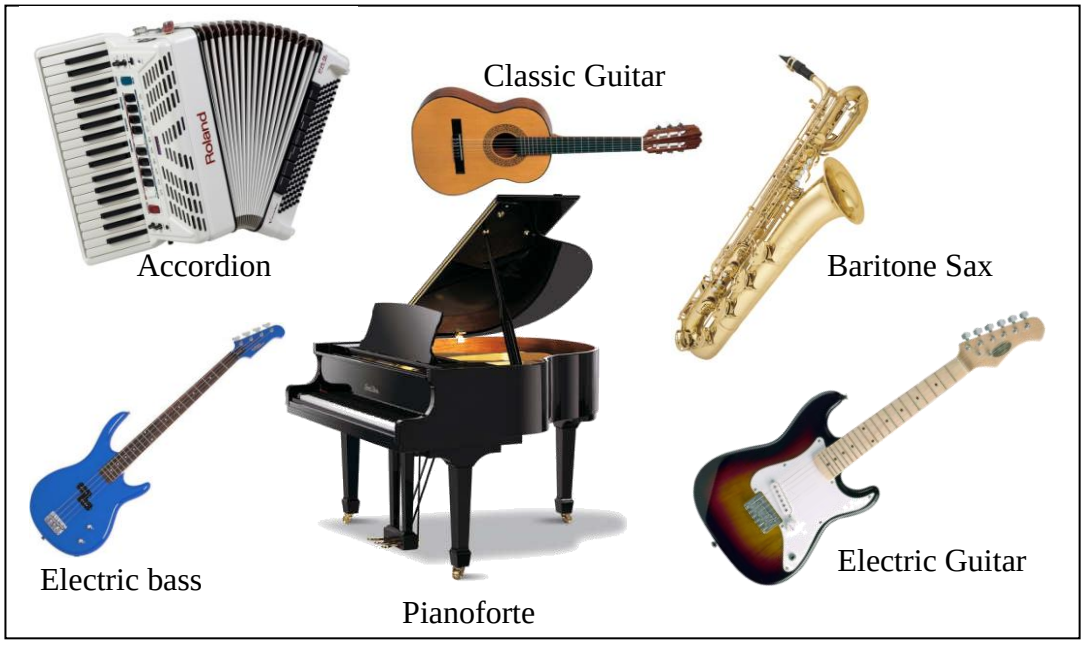


Figure 6.25 Examples of the examined instruments (accordion, Baritone Sax, classic guitar, electric bass, electric guitar, and pianoforte)

6.5. Music genres

In order to investigate whether the NZCE features apply to all types of music, various types of music were tested. The RWC genre music database contained 10 main categories:

1. popular,
2. rock,
3. dance,
4. jazz,
5. Latin,
6. classical,
7. marches,
8. world,
9. vocals, and
10. traditional Japanese music.

Each one of the 10 genres (categories)was examined individually; 3000 tracks from each genre. Each track was 3 seconds long. It was found that all musical genres lie within the same distribution of music as in the popular songs. It was found that regardless of the type of music being examined the NZCE distributions of all music genres behave in the same way.

6.6. Sampling rate

The last parameter that is investigated in this section is the effect of the sampling rate on the two novel features introduced in this thesis: the $P(NZCE_{++})$ and $P(NZCE_{--})$. It has been emphasized throughout this study to maintain the same sampling rate for both speech and music. The sampling rate is related to the number of samples per second in any audio track.

For a sampling rate of 44.1 kHz there are 44100 samples in every one second track. The number of samples in any track of certain duration, t seconds, can be computed from the sampling frequency $f_s$using the equation:

$$\text{Number of samples} = f_s \times t \tag{6.4}$$

The higher the sampling frequency used the higher the number of samples within a track of sound. For the features extracted from time series events the probability of any event is obtained by counting the number of times an event occurs and dividing by the number possible events. The latter is related to the number of samples by the equation:

$$\text{Number of events} = \text{Number of samples} - 1 \tag{6.5}$$

Furthermore, the accuracy of computing any probability of any event increases by increasing the population of events; i.e. number of samples. This is what made long tracks (e.g. 10 seconds) perform better than shorter tracks (e.g. 0.25 a second). Therefore increasing the sampling frequency is expected to increase the accuracy of computing the probability of any of time series events. The effect of the track length on the performance of each feature was examined by increasing the track length gradually from 0.25 second up to 10 seconds. In a similar manner the sampling rate is examined by down sampling the audio signals from 44.1 kHz down to 22.05 kHz, 16 kHz, 8 kHz, 4 kHz, 2 kHz, and 1 kHz. Each down-sampling is preceded by a low pass filter at half the sampling frequency. The track length is maintained at the minimum acceptable length of 3 seconds.

The $P(NZCE_{++})$ feature is sensitive to the sampling frequency. Figure 6.26 shows that resampling down to 22.05, 16, and 8 kHz is less significantly sensitive than 4, 2, and 1 kHz. This is equally true for detecting speech and music. The code scripted to analyse the effect of down-sampling is shown in listing A20.

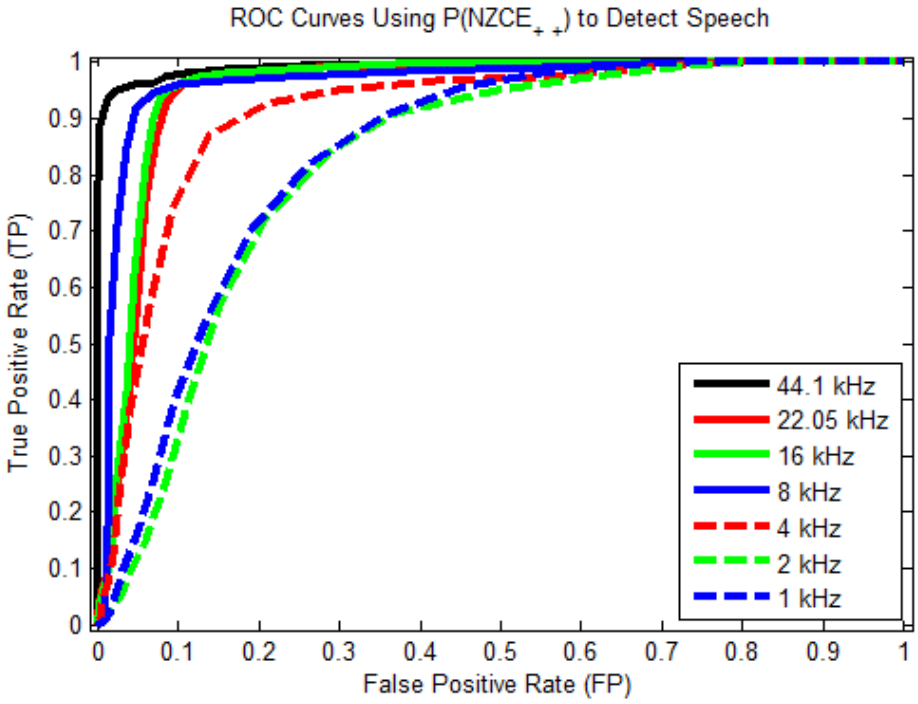
The conjugate features, P(NZCE$_{--}$) on the other hand is far less significantly sensitive to the sampling rate. Figures 6.27 and 6.28 show the ROC curves obtained when detecting speech and music, respectively.

To further demonstrate the superiority of the P(NZCE$_{--}$) over the P(NZCE$_{++}$) feature in terms of their sensitivity to the sampling frequency, figures 6.29 and 6.30 illustrate two parameters extracted from the ROC curves: the Euclidian Distance measured from the optimum TP/FP rate of each ROC curve to the optimum TP/FP rate of the ideal ROC curve.

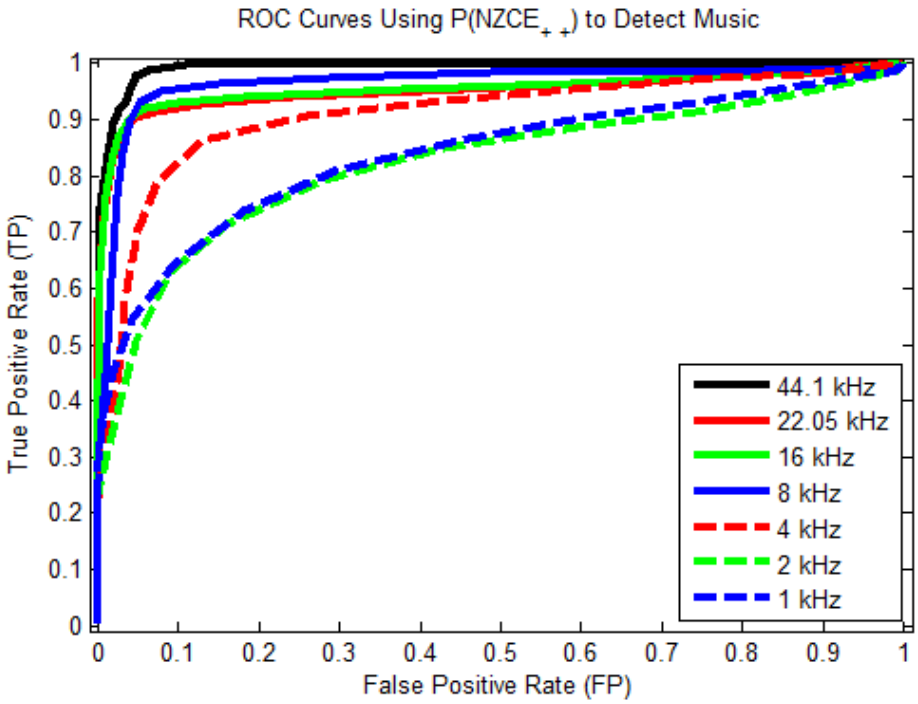
6.7. Summary

This chapter has thoroughly investigated the effect of varying the track length from ¼ of a second up to 10 seconds. It was found that the higher the track length the lower the misclassification. Likewise, the effect of down sampling from 44.1 kHz to frequencies as low as 4 kHz has significant effect on the P(NZCE$_{++}$) and insignificant effect on the P(NZCE$_{--}$).

The analysis in chapter 5 examined one type of music: the pop music. In this chapter, other types of music genres were analysed to examine the validity of the results on various types of music. Furthermore, individual instruments playing tones at an increasing pitch were also examined and verified to be consistent with all other types of music.

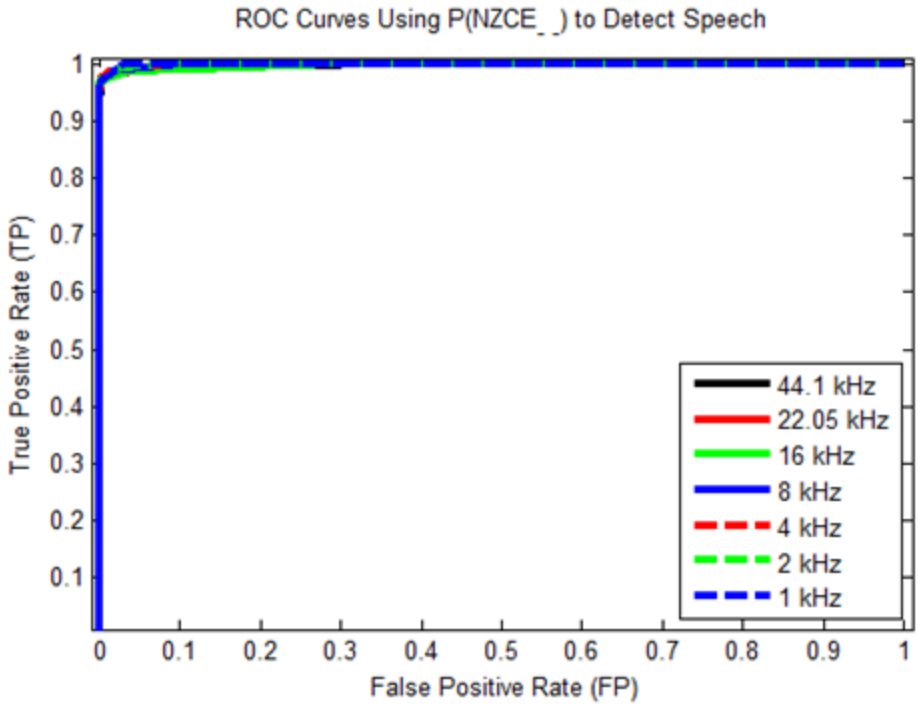


(a)

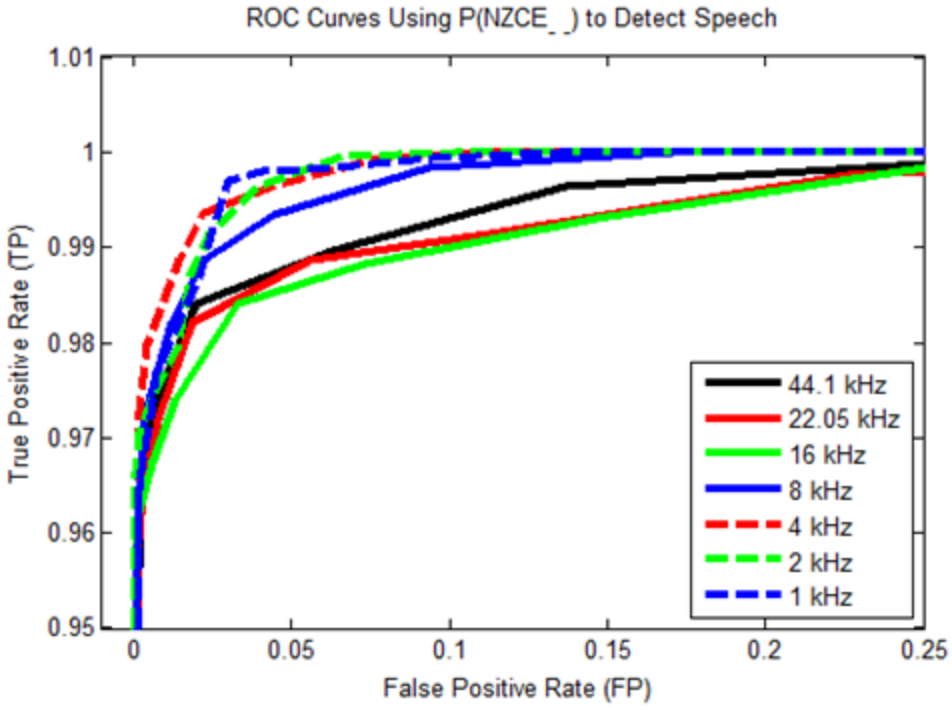


(b)

Figure 6.26 Effect of sampling frequency on speech/music detection using $P(NZCE_{+\,+})$

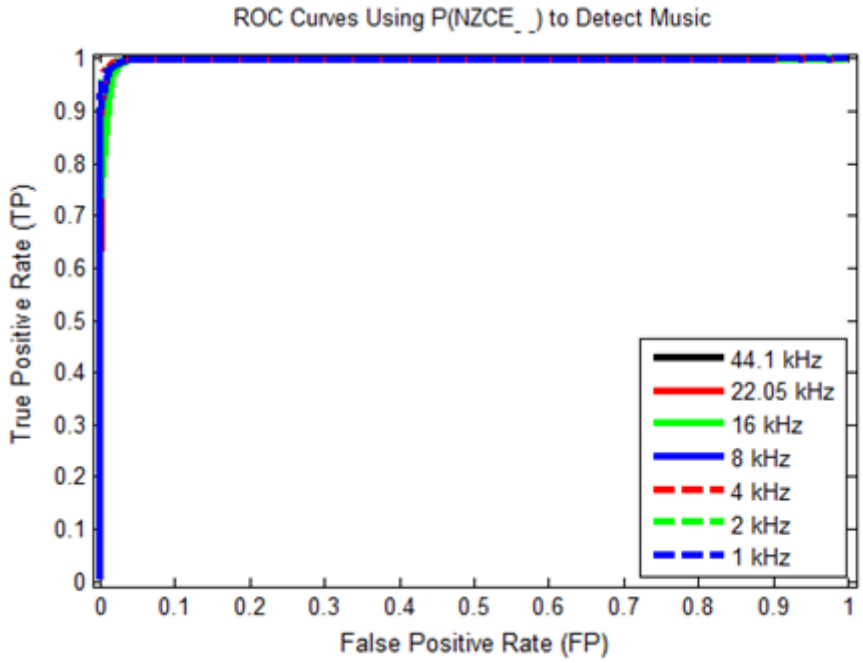


(a)

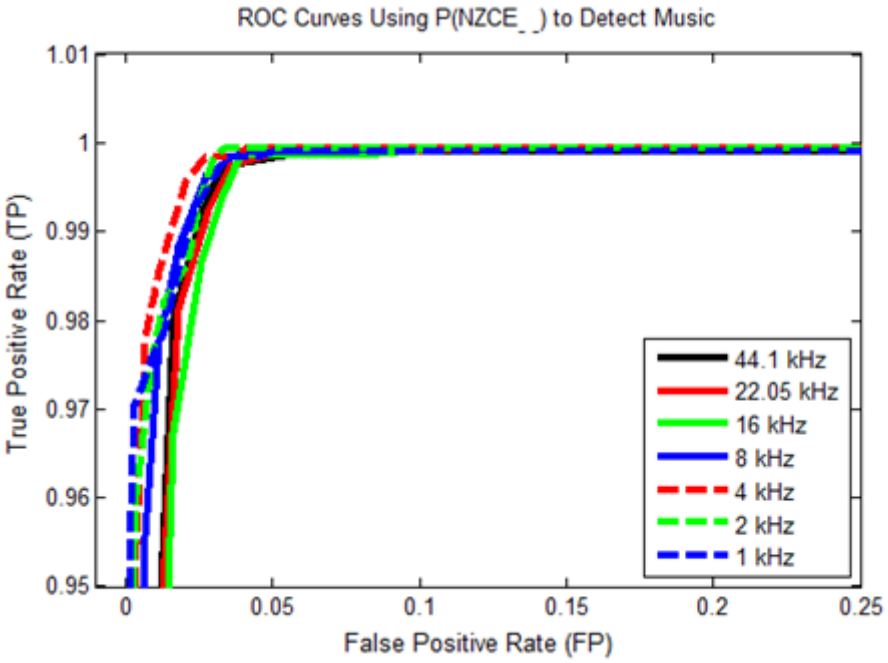


(b)

Figure 6.27 Effect of sampling frequency on speech detection using P(NZCE_ _)

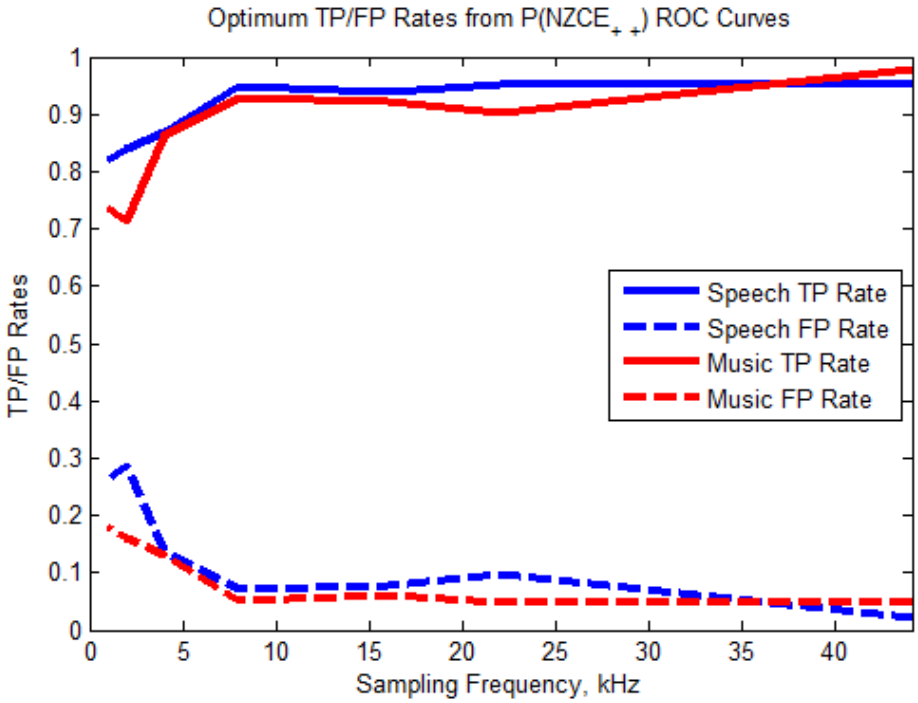


(a)

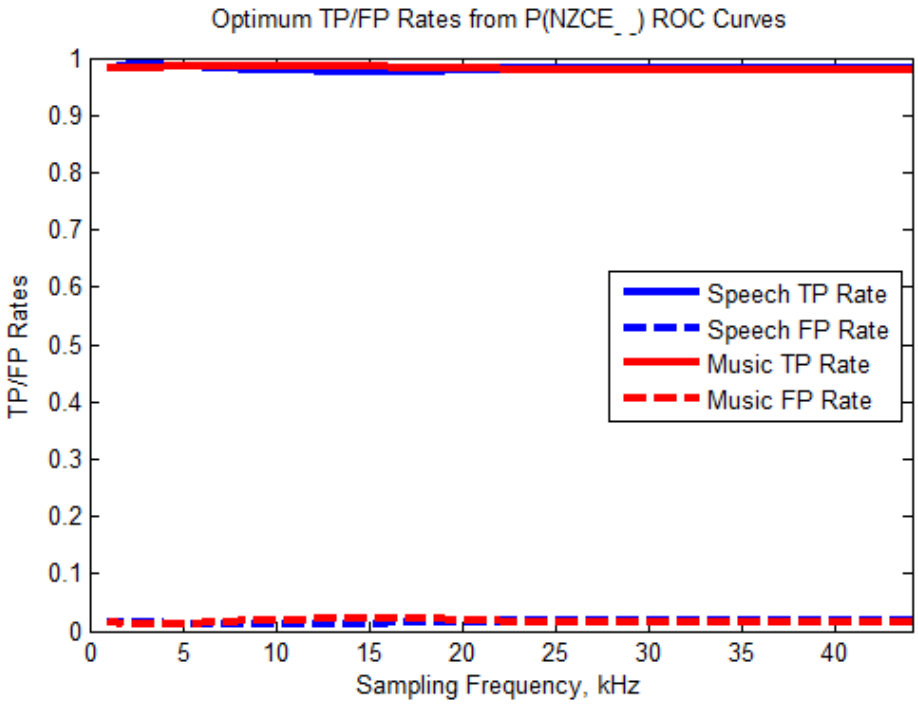


(b)

Figure 6.28 Effect of sampling frequency on music detection using P(NZCE_ _)

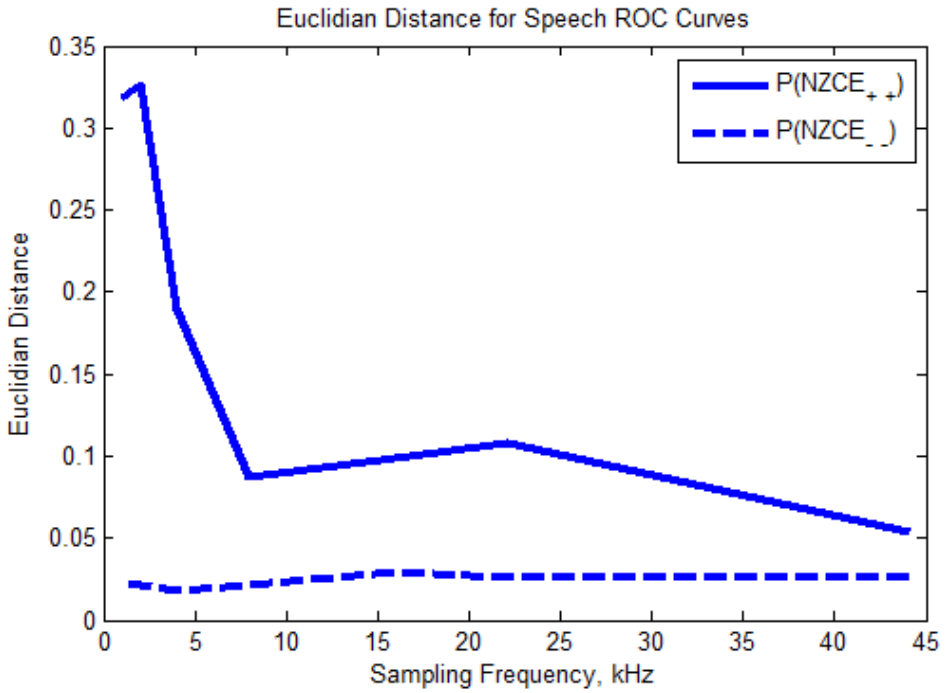


(a)

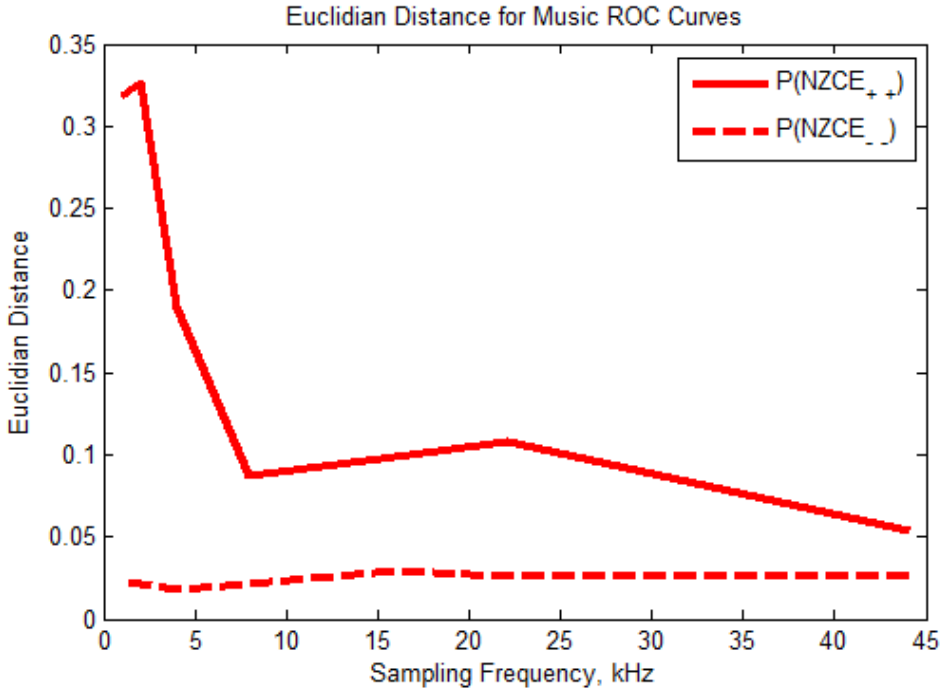


(b)

Figure 6.29 Effect of sampling frequency on the TP/FP rates when detecting speech/music when using P(NZCE$_{+\,+}$) and P(NZCE$_{-\,-}$) features



(a)



(b)

Figure 6.30 Effect of sampling frequency on the Euclidian Distance when using $P(NZCE_{+\,+})$ and $P(NZCE_{-\,-})$ features for detecting speech/music

Chapter 7: Conclusion

7.1. Introduction

Research in the field of speech/music discrimination (SMD) is continuously trying to move forward. The fundamental ingredient to any SMD system is the extraction of audio features. Whether this is done in time domain or frequency domain, any contribution is presented in the form of a novel feature or an enhancement of an existing feature.

This thesis has contributed to the SMD field of research by presenting a novel feature called the Probability of Non Zero Crossing Events, P(NZCE), and by enhancing an existing feature called the percentage of Low Energy Frames (LEF) feature. Both contributions were examined experimentally as in chapters 4, 5, and 6 by using standard data from speech and music suppliers. Various parameters were also examined. This chapter will provide a summary of the findings achieved from this study and the possible future work that could be carried forward.

7.2. Contributions

Any digital signal is composed of a series of samples that are either positive, negative, or zero. Every pair of consecutive samples forms an event. There are 9 different possible events that could occur in any digital signal. The probability of the occurrence of each type of event varies from one type to another.

After investigating all of the 9 different types of events, it was found that the best discriminating features are obtained from the Probability of Non Zero Crossing Events, P(NZCE). They are of two types:

1. consecutive positive samples; $P(NZCE_{++})$
2. consecutive negative samples; $P(NZCE_{--})$

In general, the $P(NZCE_{--})$ feature performs better than the $P(NZCE_{++})$ feature. It is not clear why the two types of NZCE show slight discrepancy in performance. Further research needs to be conducted to investigate this discrepancy.

Many parameters were investigated to examine how they would affect the performance of these two features. The following recommendations can be stated about each of them:

- Track length: should be at least 1 second. Longer tracks would provide better performance. 10 seconds track would provide error-free discrimination.
- Noise level: the higher the noise level, the higher the misclassification. Noise reduction is essential prior to using the time series events features.
- Sampling rate: higher sampling rates imply higher number of samples within an audio track; thus more accurate estimation of the probability of any event. The $P(NZCE_{++})$ is more sensitive to sampling rate than the $P(NZCE_{--})$ feature.

The thesis has also contributed another feature to the SMD field of research: the Ratio of Silent Frames (RSF) feature. The percentage of silence within an audio track is a fundamental feature that can be exploited to discriminate between speech and music. Although this exploitation is not completely new, the approach used to identify the silent frames is different. The presented approach was compared to two existing features in the literature and

was found to provide some significant enhancement. It increased the rate of correct detection of speech and music by about 2% and 8%, respectively. At the same time it reduced the misclassification of speech and music by 27% and 29%, respectively.

Another contribution that this thesis has provided to the SMD field of research is the application of the ROC curves to assess any audio feature. They provided a visual means to identify the rate of correct detection of a certain type of audio signals (e.g. speech) while also considering any misclassification due to any possible overlap with the other type (e.g. music). It is recommended to adopt this method to assess any feature suggested in the future by other researchers.

7.3. Future Work

Any research in any field would normally open doors for future contributions. This thesis has left three doors open. The first one is to investigate the effect of using classifiers other than a simple threshold. For example, clustering (as suggested in section 6.4), neural networking, and fuzzy classifiers.

The second one is related to fusion of features. The methodology adopted in this study was to present and assess individual features. Nevertheless, individual features could be combined to form a vector of features that might lead to better classifiers.

The third one is related to the hardware implementation of the proposed features. All of the proposed features were implemented by software. None of them were implemented by hardware. This work can be taken forward towards practical applications, (e.g. embedded in a radio channel scanner, embedded in an ASR system, or embedded in a hearing aid device) in order to examine its performance in real time and within a particular application. A latency of

at least 1 second is expected for acceptable performance. However, it could be made adjustable by the user based upon his/her convenience. Issues of timing, registering, sampling frequency, miniaturization, and cost efficiency could be investigated.

References

1. Al-Shoshan, A.I., *Speech and Music Classification and Separation: A Review.* Journal of King Saud University, 2006. **19**(1): p. 95-133.
2. Cardoso, J.-F., *Blind Signal Separation: Statistical Principles.* IEEE, 1998. **9**(10): p. 2009-2025.
3. Hyvärinen, A. and E. Oja, *Independent Component Analysis - Algorithms and Applications.* Neural Networks, 2000. **13**: p. 411–430.
4. Jang, G.-J., T.-W. Lee, and Y.-H. Oh, *Blind Separation of Single Channel Mixture Using ICA Basis Functions*, in *3rd International Conference on ICA and BSS (ICA2001)*. 2001: San Diego, CA, USA.
5. Li, Y. and D. Wang, *Separation of Singing Voice from Music Accompaniment for Monaural Recordings.* IEEE Transactions on Audio, Speech, and Language Processing, 2007. **15**(4): p. 1475-1487.
6. Murata, N., S. Ikeda, and A. Ziehe, *An Approach to Blind Source Separation Based on Temporal Structure of Speech Signals.* Neurocomputing, 2001. **41**: p. 1-24.
7. Schobben, D., K. Torkkola, and P. Smaragdis, *Evaluation of Blind Signal Separation Methods*, in *First International Workshop on Independent Component Analysis and Blind Signal Separation*. 1999: Aussois, France.
8. Singh, Y. and C.S. Rai, *Blind Source Separation: A Unified Approach.* Neurocomputing, 2002. **49**: p. 435 – 438.
9. Torkkola, K. *Blind Separation for Audio Signals - Are We There Yet?* in *Proc. Workshop on Independent Component Analysis and Blind Signal Separation*. 1999. Aussois, France.

10. Wang, D.L. and G.J. Brown, *Separation of Speech from Interfering Sounds Based on Oscillatory Correlation.* IEEE Transactions on Neural Networks, 1999. **10**(3): p. 684-697.

11. Alnadabi, M. and S. Johnstone. *Discrimination Between Speech and Music Using Time Series Events*. in *ICSP08*. 2008. Beijing, China: IEEE.

12. Alnadabi, M. and S. Johnstone, *Speech/music Discrimination by Detection: Assessment of Time Series Events Using ROC Graphs*, in *Systems, Signals and Devices, SSD'09*. 2009, IEEE: Djerba, Tunisia. p. 1-5.

13. Carey, M.J., E.S. Parris, and H. Lloyd-Thomas. *A Comparison of Features for Speech, Music Discrimination*. in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'99)*. 1999.

14. El-Maleh, K., et al. *Speech/Music Discrimination for Multimedia Applications*. in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP '2000)*. 2000.

15. Jarina, R., et al., *Speech-Music Discrimination from MPEG-1 Bitstream*, in *SSIP 2001 - WSES International Conference on Speech, Signal and Image Processing*. 2001: Malta. p. 1-6.

16. Kedem, B., *Spectral Analysis and Discrimination by Zero-Crossings.* Proceedings of the IEEE, 1986. **74**(11): p. 1477-1493.

17. Mesgarani, N., *Discrimination of Speech from Non-Speech Based on Multiscale Spectro-Temporal Modulations*, in *Department of Electrical and Computer Engineering*, . 2005, University of Maryland: Maryland, USA. p. 23.

18. Mesgarani, N., M. Slaney, and S.A. Shamma, *Discrimination of Speech from Nonspeech Based on Multiscale Spectro-Temporal Modulations.* IEEE Transactions on Audio, Speech, and Language Processing, 2006. **14**(3): p. 920-930.

19. Munoz-Exposito, J.E., et al., *Adaptive Network-Based Fuzzy Inference System vs. other Classification Algorithms for Warped LPC-Based Speech/Music Discrimination.* Engineering Application of Artificial Intelligence, 2007. **20**: p. 783-793.

20. Ntalampiras, S. and N. Fakotakis, *Speech/Music Discrimination Based on Discrete Wavelet Transform*, in *Lecture Notes in Computer Science (LNCS)*. 2008, Springer Berlin: Heidelberg. p. 205-211.

21. Saad, E.M., et al. *A Multifeature Speech/Music Discrimination System*. in *Nineteenth National Radio Science Conference*. 2002. Alexandria: URSI.

22. Saunders, J. *Real-Time Discrimination of Broadcast Speech-Music*. in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'96)*. 1996. Toulouse.

23. Wang, W.Q., W. GaO, and D.W. Ying. *A Fast and Robust Speech/Music Discrimination Approach*. in *ICICS-PCM 2003*. 2003. Singapore.

24. Zhou, H., A. Sadka, and R.M. Jiang. *Feature Extraction for Speech and Music Discrimination*. in *The Sixth International Workshop on Content-Based Multimedia Indexing*. 2008. London, UK: IEEE.

25. Harb, H., L. Chen, and J.-Y. Auloge. *Speech/Music/Silence and Gender Detection Algorithm*. in *Proceedings of the 7th International conference on Distributed Multimedia Systems DMS01*. 2001. Taipei, Taiwan.

26. Essid, S., G. Richard, and B. David, *Musical Instrument Recognition by Pairwise Classification Strategies.* IEEE Transactions on Audio, Speech, and Language Processing, 2006. **14**(4): p. 1401-1412.

27. Kitahara, T., M. Goto, and H.G. Okuno, *Pitch-dependent Identification of Musical Instrument Sounds.* Applied Intelligence(The International Journal of Artificial

Intelligence, Neural Networks, and Complex Problem-Solving Technologies), 2005. **23**(3): p. 267-275.

28. Pikrakis, A., S. Theodoridis, and D. Kamarotos, *Classification of Musical Patterns Using Variable Duration Hidden Markov Models.* IEEE Transactions on Audio, Speech, and Language Processing, 2006. **14**(5): p. 1795-1807.
29. Tzanetakis, G. and P. Cook, *Musical Genre Classification of Audio Signals.* IEEE Transactions on Speech and Audio Processing, 2002. **10**(5): p. 293-302.
30. Scheirer, E. and M. Slaney. *Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator*. in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97)*. 1997.
31. Panagiotakis, C. and G. Tziritas, *A Speech-Music Discriminator Based on RMS and Zero-Crossings.* IEEE Transactions on Multimedia, 2005. **7**(1): p. 155-166.
32. Wang, Y.-P., et al., *Fast Frequency Estimation by Zero Crossings of Differential Spline Wavelet Transform.* EURASIP Journal on Applied Signal Processing, 2005. **8**: p. 1251-1260.
33. Chasin, M., *Music & Hearing Aids*, in *La Scena Musicale*. 2007.
34. Becker, T.J., *Understanding Decibel Levels and Hazards*, in *Chicago Tribune*. 1996.
35. Nordqvist, P. and A. Leijon, *An Efficient Robust Sound Classification Algorithm for Hearing Aids.* Journal of Acoustical Society of America, 2004. **115**(6): p. 3033–3041.
36. Mahdi, W., M. Ardebilian, and L. Chen, *Automatic Scene Segmentation Based on Spatial-Temporal Clues and Rhythm.* International Journal of Networking and Information Systems, 2001. **5**.
37. Pye, D., et al. *Audio Visual Segmentation for Content Based Retrieval*. in *International conference on spoken language processing (ICSLP 98)*. 1998. Sydney, Australia.

38. Gauch, J.M., et al., *Real Time Video Scene Detection and Classification.* Information Processing and Management, 1999(35): p. 401-420.

39. Wang, Y., Z. Liu, and J.-C. Huang, *Multimedia Content Analysis: Using Both Audio and Visual Clues*, in *IEEE Signal Processing Magazine*. 2000, IEEE. p. 12-36.

40. MARKHAM, D. and V. HAZAN, *Speech, Hearing and Language: Work in Progress.* The UCL Speaker Database, 2002. **14**: p. 1-17.

41. Goto, M. *Development of the RWC Music Database*. in *Proceedings of the 18th International Congress on Acoustics (ICA 2004)*. 2004.

42. Goto, M., et al. *RWC Music Database: Popular, Classical, and Jazz Music Databases*. in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*. 2002.

43. Goto, M., et al. *RWC Music Database: Music Genre Database and Musical Instrument Sound Database*. in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*. 2003.

44. Zolzer, U., *Digital Audio Signal Processing, Second Edition*. 2008, Hamburg, Germany: Helmut Schmidt University.

45. Owens, F.J., *Signal Processing of Speech*. 1993: McGraw-Hill.

46. Proakis, J.G. and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. Third ed. 2000, New Delhi: Prentice-Hall India.

47. Gold, B. and L.R. Rabiner, *Parallel Processing Techniques for Estimating Pitch Periods of Speech in The Time-Domain.* Journal of Acoustical Society of America, 1969. **46**: p. 442-448.

48. Markel, J.D., *The SIFT Algorithm for Fundamental Frequency Estimation.* IEEE Transactions on Audio Electroacoustics, 1972. **AU-20**: p. 367-377.

49. Hess, W., *Pitch Determination of Speech Signals*. Vol. Springer-Verlag. 1983.

50. Liu, Z., Y. Wang, and T. CHEN, *Audio Feature Extraction and Analysis for Scene Segmentation and Classification.* Journal of VLSI Signal Processing, 1998. **20**: p. 61–79.

51. Ishizuka, K. and N. Miyazaki, *Speech Feature Extraction Method Using Subband-Based Periodicity and Nonperiodicity Decomposition.* Journal of Acoustical Society of America, 2006. **120**(1): p. 443–452.

52. Mauclair, J. and J. Pinquier. *Fusion of Descriptors for Speech Music Classification*. in *12th European Signal Processing Conference (EUSIPCO-2004).* 2004. Vienna, Austria.

53. http://www.soundfisher.com. *Sound Fisher Software, Muscle Fish - a Division of Audible Magic Inc., 2550 Ninth St., Suite 207B Berkeley, CA 94710*

54. Fawcett, T., *ROC Graphs: Notes and Practical Considerations for Researchers*, in *HP Laboratories*, H. Laboratories, Editor. 2004.

55. Fawcett, T., *An Introduction to ROC Analysis.* Pattern Recognition Letters, 2006 **27**: p. 861-874.

56. Lu, L., H. Jiang, and H. Zhang, *A Robust Audio Classification and Segmentation Method*, in *Ninth ACM international conference on Multimedia* 2001, ACM: Ottawa, Canada. p. 203-211.

57. Standard, A.N., *ATIS Telecom Glossary*, in *ATIS Standards*. 2007.

58. Brown, R.G., *Introduction to Random Signal Analysis and Kalman Filtering*. 1983: John Wiley and Sons.

59. Papoulis, A., *Probability, Random Variables, and Stochastic Processes, Third edition*. 1991: WCB/McGraw-Hill.

60. Keshner, M., *1/f Noise.* Proceedings of IEEE, 1982. **70**: p. 212-218.

61. Ro, W. and Y. Kwon, *1/f Noise Analysis of Songs in Various Genre of Music.* Chaos, Solitons & Fractals, 2009.

62. Voss, R. and J. Clarke, *1/f Noise in Music and Speech.* Nature 1975: p. 258-317.

63. Voss, R. and J. Clarke, *1/f Noise in Music.* Journal of Acoustical Society of America, 1978: p. 63-258.

Appendix A: MATLAB Codes

Listing A1

Code scripted as a function that computes the parameters of a ROC curve of any pair of distributions where distribution a is lower than distribution b

```
% This function takes two inputs (a,b) and computes the ROC curve
% parameters as explained below:
%
% FP:       Vector of False positive rates
% TP:       Vector of True positive rates
% Distance: Eucledian Distance from (a,b) to (0,1) at current threshold
% opt_TP:   Optimum True Positive Rate
% opt_FP:   Optimum False Positive Rate
% opt_thr:  Optimum threshold for lowest Eucledian Distance from (a,b) to
(0,1)


function [FP,TP,Distance,opt_TP,opt_FP,opt_thr] = roc(a, b)

% Initial values declared

          n = 0;              % Initial counter
          N = length(a);      % Number of samples in class (a)
         FP = zeros(N-1,1);   % Vector of FP rates
         TP = zeros(N-1,1);   % Vector of TP rates
upper_limit = 1;              % Inital upper limit of samples
   Distance = 1e6;            % Initial Distance set to large value

% Scan the whole range

while upper_limit <= N - 1
    n = n + 1;
    TP(n,1) = sum(a(1:upper_limit))/sum(a);         % Compute TP
    FP(n,1) = sum(b(1:upper_limit))/sum(b);         % Compute FP
          upper_limit = upper_limit + 1;            % Increment upper_limit
temp_Distance = sqrt((TP(n,1)-1)^2+(FP(n,1)-0)^2); % Eucledian Distance
if temp_Distance < Distance                       % Check for lower distance
    Distance = temp_Distance;                       % Update Distance
      opt_TP = TP(n,1);                             % Update Optimum TP rate
      opt_FP = FP(n,1);                             % Update Optimum FP rate
     opt_thr = upper_limit/N;                       % Update Optimum
threshold
end
end
```

Listing A2

Code scripted as a function that computes the parameters of a ROC curve of any pair of distributions where distribution b is lower than distribution a

```
% This function takes two inputs (a,b) and computes the ROC curve
% parameters as explained below:
%
% FP:       Vector of False positive rates
% TP:       Vector of True positive rates
% Distance: Eucledian Distance from (a,b) to (0,1) at current threshold
% opt_TP:   Optimum True Positive Rate
% opt_FP:   Optimum False Positive Rate
% opt_thr:  Optimum threshold for lowest Eucledian Distance from (a,b) to
(0,1)


function [FP,TP,Distance,opt_TP,opt_FP,opt_thr] = roc2(a, b)

% Initial values declared

          n = 0;            % Initial counter
          N = length(a);    % Number of samples in class (a)
         FP = zeros(N-1,1); % Vector of FP rates
         TP = zeros(N-1,1); % Vector of TP rates
lower_limit = N;            % Inital lower limit of samples
   Distance = 1e6;          % Initial Distance set to large value

% Scan the whole range

while lower_limit >= 1
    n = n + 1;
    TP(n,1) = sum(a(lower_limit: N))/sum(a);       % Compute TP
    FP(n,1) = sum(b(lower_limit: N))/sum(b);       % Compute FP
          lower_limit = lower_limit - 1;           % Increment upper_limit
temp_Distance = sqrt((TP(n,1)-1)^2+(FP(n,1)-0)^2); % Eucledian Distance
if temp_Distance < Distance                      % Check for lower distance
    Distance = temp_Distance;                        % Update Distance
      opt_TP = TP(n,1);                              % Update Optimum TP rate
      opt_FP = FP(n,1);                              % Update Optimum FP rate
     opt_thr = lower_limit/N;                        % Update Optimum
threshold
end
end
```

Listing A3

Code scripted as a function that computes the RMS of any input signal

```
function power = rms(x)

power = sqrt(sum(x.^2)/length(x));
```

Listing A4

Code scripted as a function to compute the energy of any input signal

```
function ENERGY = energy(signal)

ENERGY = sum(signal.^2);
```

Listing A5

Code scripted as a function to compute the LEF feature of any input signal

```
% This function splits the input signal into frames
% and computes the RMS for each frame;
% the percentage of frames having RMS above the mean RMS is computed.

function LOW_ENERGY_FRAMES = LEF(signal, fs, split)

size = length(signal);              % length of the input signal

sln = floor(split * fs);            % segment length = split second(s)

segments = floor(size/sln);         % Number of segments

RMS = zeros(segments,1);            % Reserve a zero vector

for r = 1 : segments
        segment = signal((r-1)*sln+1:r*sln);
       RMS(r,1) = rms(segment);
end

threshold = 0.5 * mean(RMS);

LOW_ENERGY_FRAMES = length(find(RMS<threshold))/length(RMS)*100;
```

Listing A6

Code scripted as a function to compute the MLER feature of any input signal

```
% This function splits the input signal into frames
% and computes the ENERGY for each frame;
% the percentage of frames having ENERGY above the mean RMS is computed.

function MODIFIED_LOW_ENERGY_RATIO = MLER(signal, fs, split, delta)

size = length(signal);              % length of the input signal
sln = floor(split * fs);            % segment length = split second(s)
segments = floor(size/sln);         % Number of segments
ENERGY = zeros(segments,1);         % Reserve a zero vector

for r = 1 : segments
        segment = signal((r-1)*sln+1:r*sln);
    ENERGY(r,1) = energy(segment);
end

threshold = delta * mean(ENERGY);

MODIFIED_LOW_ENERGY_RATIO = ...
length(find(ENERGY<threshold))/length(ENERGY)*100;
```

Listing A7

Code scripted as a function to compute the RSF feature of any input signal

```
% This function computes the RSF feature of any input signal

functionRATIO_OF_SILENT_Frames = RSF(signal, fs, split, delta)

         N = length(signal);      % length of the input signal
       sln = floor(split * fs);   % segment length = split second(s)
  segments = floor(N/sln);        % Number of segments
ENERGYxZCR = zeros(segments,1);   % Reserve a zero vector

for r = 1 : segments
        segment = signal((r-1)*sln+1:r*sln);
ENERGYxZCR(r,1) = energy(segment)*zcrPT(segment);

end

thresh = delta * mean(ENERGYxZCR);
RATIO_OF_SILENT_Frames = ...
length(find(ENERGYxZCR<thresh))/length(ENERGYxZCR)*100;
```

Listing A8

Code scripted to analyse the application of LEF feature for speech/music discrimination

```
% The Low Energy Frames (LEF) feature is used to analyze speech and music
% for comparison.


%% Declare constants
clear all
            fs = 44100;
 target_tracks = 3000;
             r = 1; % accumulated stream of sound
         split = 20e-3;
     LEFspeech = [];
      LEFmusic = [];

 speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword';
  music_folder = 'C:\DATA\MUSIC\pop_music\popmusic';

speech_counter = 0;
 music_counter = 0;

%% Obtain LEF for each frame
analyzed_tracks = 0;

while analyzed_tracks < target_tracks

% Analyze speech signals

duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
     speech = wavread([speech_folder,num2str(speech_counter)]);
     speech = speech(:,1);
    speech_ = [speech_; speech];
   duration = length(speech_);
end

   speech = speech_(1:r*fs);
   speech = speech-mean(speech);
   speech = speech/max(abs(speech));
LEFspeech = [LEFspeech; LEF(speech, fs, split)];
```

Listing A8 (Continued ...)

```
% Analyze pop music signals

duration = 0;
  music_ = [];
while duration < r*fs
if music_counter >= 4094
        music_counter = 0;
end
music_counter  = music_counter + 1;
     music = wavread([music_folder,num2str(music_counter)]);
     music = music(:,1);
    music_ = [music_; music];
   duration = length(music_);
end

   music = music_(1:r*fs);
   music = music-mean(music);
   music = music/max(abs(music));
LEFmusic = [LEFmusic; LEF(music, fs, split)];

     A1 = length(LEFspeech);
     A2 = length(LEFmusic);

analyzed_tracks = min([A1, A2]);
end

LEFspeech = LEFspeech(1:analyzed_tracks);
 LEFmusic = LEFmusic(1:analyzed_tracks);

save LEF_results

%% %%%%%%%%%%%%%%%%%%%%%%%% LEF Distributions  %%%%%%%%%%%%%%%%%%%%%%%%%%%

Bins = 1 : 4 : 100;

[LEFspeechY, LEFspeechX] = rhist(LEFspeech, Bins);
[LEFmusicY,  LEFmusicX ] = rhist(LEFmusic, Bins);

LEFspeechY = 100 * LEFspeechY;
 LEFmusicY = 100 * LEFmusicY;

[FP_speech_LEF,TP_speech_LEF,Distance_speech_LEF,optimum_TP_speech_LEF,...
 optimum_FP_speech_LEF,speech_LEF_threshold] = roc2(LEFspeechY, LEFmusicY);
[FP_music_LEF,TP_music_LEF,Distance_music_LEF,optimum_TP_music_LEF,...
    optimum_FP_music_LEF,music_LEF_threshold] = roc(LEFmusicY, LEFspeechY);
```

Listing A8 (Continued ...)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot Distributions %%%%%%%%%%%%%%%%%%%%%%

figure('color','white'); hold on
plot(LEFspeechX, LEFspeechY, 'b', 'LineWidth', 3)
plot(LEFmusicX,  LEFmusicY,  'r', 'LineWidth', 3)
title('Distribution of the LEF feature')
xlabel('Percentage of Low Energy Frames (LEF), %')
ylabel('Percentage of tracks, %')
legend('Speech', 'Music')
xlim([0 100])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot ROC Curves %%%%%%%%%%%%%%%%%%%%%%%%%%%

figure('color','white')
xlim([0 1]), ylim([0 1])
plot(FP_speech_LEF, TP_speech_LEF, 'b', FP_music_LEF, TP_music_LEF, 'r',...
'Linewidth', 3)
xlabel(['False Positive (FP) Rate'])
ylabel(['True Positive (TP) Rate'])
title(['ROC Curves for LEF Feature'])
legend('Speech detection', 'Music detection')
```

Listing A9

Code scripted to analyze the application of MLER feature for speech/music discrimination

```
% The Modified Low Energy Ratio (MLER) feature is used to analyze speech
% and music for comparison; delta used = 0.1

%% Declare constants
clear all
            fs = 44100;
 target_tracks = 3000;
             r = 1; % accumulated stream of sound
         split = 20e-3;
    MLERspeech = [];
     MLERmusic = [];
         delta = 0.1;

 speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword';
  music_folder = 'C:\DATA\MUSIC\pop_music\popmusic';

speech_counter = 0;
 music_counter = 0;

%% Obtain MLER for each frame
analyzed_tracks = 0;
while analyzed_tracks < target_tracks


% Analyze speech signals
duration = 0;
 speech_ = [];
while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
     speech = wavread([speech_folder,num2str(speech_counter)]);
     speech = speech(:,1);
    speech_ = [speech_; speech];
   duration = length(speech_);
end
   speech = speech_(1:r*fs);
   speech = speech-mean(speech);
   speech = speech/max(abs(speech));
MLERspeech = [MLERspeech; MLER(speech, fs, split, delta)];
```

Listing A9 (Continued ...)

```
% Analyze pop music signals

duration = 0;
  music_ = [];
while duration < r*fs
if music_counter >= 4094
        music_counter = 0;
end

music_counter  = music_counter + 1;
     music = wavread([music_folder,num2str(music_counter)]);
     music = music(:,1);
    music_ = [music_; music];
   duration = length(music_);
end

   music = music_(1:r*fs);
   music = music-mean(music);
   music = music/max(abs(music));
MLERmusic = [MLERmusic; MLER(music, fs, split, delta)];

     A1 = length(MLERspeech);
     A2 = length(MLERmusic);

analyzed_tracks = min([A1, A2]);

end

MLERspeech = MLERspeech(1:analyzed_tracks);
 MLERmusic = MLERmusic(1:analyzed_tracks);

save MLER_results

%% %%%%%%%%%%%%%%%%%%%%%%%%% MLER Distributions %%%%%%%%%%%%%%%%%%%%%%%%%
Bins = 1 : 4 : 100;

[MLERspeechY, MLERspeechX] = rhist(MLERspeech, Bins);
[MLERmusicY,  MLERmusicX ] = rhist(MLERmusic, Bins);

MLERspeechY = 100 * MLERspeechY;
 MLERmusicY = 100 * MLERmusicY;

[FP_speech_MLER,TP_speech_MLER,Distance_speech_MLER,...
optimum_TP_speech_MLER,optimum_FP_speech_MLER,speech_MLER_threshold] = ...
                                                 roc2(MLERspeechY, MLERmusicY);
[FP_music_MLER,TP_music_MLER,Distance_music_MLER,...
optimum_TP_music_MLER,optimum_FP_music_MLER,music_MLER_threshold] = ...
                                                  roc(MLERmusicY, MLERspeechY);
```

Listing A9 (Continued ...)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot Distributions %%%%%%%%%%%%%%%%%%%%%%%
figure('color','white'); hold on
plot(MLERspeechX, MLERspeechY, 'b', 'LineWidth', 3)
plot(MLERmusicX,  MLERmusicY,  'r', 'LineWidth', 3)
title('Distribution of the MLER feature')
xlabel('Modified Low Energy Ratio (MLER), %')
ylabel('Percentage of tracks, %')
legend('Speech', 'Music')
xlim([0 100])

%%%%%%%%%%%%%%%%%%%%%%% Plot ROC Curves %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure('color','white')
xlim([0 1]), ylim([0 1])
plot(FP_speech_MLER, TP_speech_MLER, 'b', FP_music_MLER, TP_music_MLER,
'r', 'Linewidth', 3)
xlabel(['False Positive (FP) Rate'])
ylabel(['True Positive (TP) Rate'])
title(['ROC Curves for MLER Feature'])
legend('Speech detection', 'Music detection')
```

Listing A10

Code scripted to analyze the application of RSF feature for speech/music discrimination

```
% The Ratio of Silent Frames (RSF) feature is used to analyze speech and
% music for comparison;

%% Declare constants
            fs = 44100;
 target_tracks = 3000;
             r = 1;                                % accumulated stream of sound
         split = 20e-3;
    RSFspeech = zeros(target_tracks,1);
     RSFmusic = zeros(target_tracks,1);
         delta = 0.1;

 speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword';
  music_folder = 'C:\DATA\MUSIC\pop_music\popmusic';

speech_counter = 0;
 music_counter = 0;

%% Obtain RSF for each track
for track = 1 : target_tracks
% Analyze speech signals
duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
     speech = wavread([speech_folder,num2str(speech_counter)]);
     speech = speech(:,1);
    speech_ = [speech_; speech];
   duration = length(speech_);
end

   speech = speech_(1:r*fs);
   speech = speech-mean(speech);
   speech = speech/max(abs(speech));
RSFspeech(track)= RSF(speech, fs, split, delta);

% Analyze pop music signals
duration = 0;
  music_ = [];

while duration < r*fs
if music_counter >= 4094
        music_counter = 0;
end
music_counter  = music_counter + 1;
     music = wavread([music_folder,num2str(music_counter)]);
     music = music(:,1);
    music_ = [music_; music];
   duration = length(music_);
end
```

Listing A10 (Continued ...)

```
   music = music_(1:r*fs);
   music = music-mean(music);
   music = music/max(abs(music));
RSFmusic(track)= RSF(music, fs, split, delta);

end

save RSF_results

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% RSF %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Bins = 1 : 4 : 100;

[RSFspeechY, RSFspeechX] = rhist(RSFspeech, Bins);
[RSFmusicY,  RSFmusicX ] = rhist(RSFmusic,  Bins);

RSFspeechY = 100 * RSFspeechY;
 RSFmusicY = 100 * RSFmusicY;

[FP_speech_RSF,TP_speech_RSF,Distance_speech,...
    optimum_TP_speech_RSF,optimum_FP_speech_RSF,speech_RSF_threshold] = ...
                                                    roc2(RSFspeechY, RSFmusicY);
[FP_music_RSF,TP_music_RSF,Distance_music_RSF,...
       optimum_TP_music_RSF,optimum_FP_music_RSF,music_RSF_threshold] = ...
                                                     roc(RSFmusicY, RSFspeechY);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot distributions %%%%%%%%%%%%%%%%%%%%%%%%%%%
figure('color','white'); hold on
plot(RSFspeechX, RSFspeechY, 'b', 'LineWidth', 3)
plot(RSFmusicX,  RSFmusicY,  'r', 'LineWidth', 3)
title('Distribution of the RSF feature)
xlabel('Ratio of Silent Frames (RSF), %')
ylabel('Percentage of tracks, %')
legend('Speech', 'Music')
xlim([0 100])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot ROC Curves %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure('color','white')
xlim([0 1]), ylim([0 1])
plot(FP_speech_RSF, TP_speech_RSF, 'b', FP_music_RSF, TP_music_RSF, 'r',
'Linewidth', 3)
xlabel(['False Positive (FP) Rate'])
ylabel(['True Positive (TP) Rate'])
title(['ROC Curves for RSF Feature using Energy'])
legend('Speech detection', 'Music detection')
```

Listing A11

Code scripted as a function to be used for computing the Zero Crossing Rate (ZCR) of any input signal

```
% This function computes the zcr using the definition proposed by:
% Panagiotakis and Tziritas

function count = zcrPT(x)

signs = abs(sign(x(2:length(x))) - sign(x(1:length(x)-1)));

N = length(x)-1;

count = 0.5 * sum(signs) / N ;
```

Listing A12

Code scripted as a function to compute the probability of the occurrence of all possible events in any given input signal

```
% This function computes the probabilities of each of the 9 possible events
% that could occur in an audio signal. The events are as illustrated below:
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%E1) ---x---x--- zero then zero                                              %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            x                                                                %
%E2) ---x------- zero then positive                                          %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%E3) ---x------- zero then negative                                          %
%            x                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        x                                                                    %
%E4) -------x--- positive then zero                                          %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%E5) -------x--- negative then zero                                          %
%        x                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            x                                                                %
%        x                                                                    %
%E6) ----------- positive then positive                                      %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%E7) ----------- negative then negative                                      %
%       x                                                                     %
%            x                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        x                                                                    %
%E8) ----------- positive then negative                                      %
%            x                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            x                                                                %
%E9) ----------- negative then positive                                      %
%        x                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Probabilities = events_probabilities(signal)

N = length(signal)-1;
```

Listing A12 (Continued ...)

```
  E1 = 0; % 0 0
  E2 = 0; % 0 +
  E3 = 0; % 0 -
  E4 = 0; % + 0
  E5 = 0; % - 0
  E6 = 0; % + +
  E7 = 0; % - -
  E8 = 0; % + -
  E9 = 0; % - +
E167 = 0; % nzcr count = E1 or E6 or E7
 E89 = 0; %  zcr count = E8 or E9

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for r = 1 : N
if signal(r) == 0  & signal(r+1) == 0    % 0 0
        E1 = E1   + 1;
      E167 = E167 + 1;
elseif signal(r) == 0  & signal(r+1) >  0    % 0 +
        E2 = E2 + 1;
elseif signal(r) == 0  & signal(r+1) <  0    % 0 -
        E3 = E3 + 1;
elseif signal(r) >  0  & signal(r+1) == 0    % + 0
        E4 = E4 + 1;
elseif signal(r) <  0  & signal(r+1) == 0    % - 0
        E5 = E5 + 1;
elseif signal(r) >  0  & signal(r+1) >  0    % + +
        E6 = E6   + 1;
      E167 = E167 + 1;
elseif signal(r) <  0  & signal(r+1) <  0    % - -
        E7 = E7   + 1;
      E167 = E167 + 1;
elseif signal(r) >  0  & signal(r+1) <  0    % + -
        E8 = E8   + 1;
       E89 = E89  + 1;
elseif signal(r) <  0  & signal(r+1) >  0    % - +
        E9 = E9   + 1;
       E89 = E89  + 1;
end
end

% Divide by the number of events to compute the probability

Probabilities = [E1, E2, E3, E4, E5, E6, E7, E8, E9, E167, E89]/N;
```

Listing A13

Code scripted as a function to be used for computing the probability of the occurrence of the two types of Non-Zero Crossing Events (NZCE) for any input signal

```
% This function computes the probability of the NZCE for the two types:
% positive and negative.

function [P_NZCE_pos, P_NZCE_neg] = P_NZCE(input)

N = length(input);

positive = zeros(1,N);
negative = zeros(1,N);

positive(find(input>0)) = 1;
negative(find(input<0)) = 1;

count_positive = sum(positive(2:N).*positive(1:N-1));
P_NZCE_pos = count_positive/(N-1);

count_negative = sum(negative(2:N).*negative(1:N-1));
P_NZCE_neg = count_negative/(N-1);
```

Listing A14

Code scripted as a function to be used for computing the probability of the occurrence of the two types of Non-Zero Crossing Events (NZCE) for any input signal, after splitting the input into tracks of certain duration of seconds.

```
% This function splits the input signal into tracks of "split" seconds
% each and computes the probability of each event for each track and
% returns a vector of probabilities ( 9 individuals + 2 sets)

function P = events_probabilities_per_track(signal,fs,split)

      size = length(signal);              % length of the input signal

       sln = floor(split * fs);           % segment length = split seconds

  segments = floor(size/sln);             % Number of segments

         P = zeros(segments,11);          % Reserve a zero vector

for r = 1 : segments
    segment = signal((r-1)*sln+1:r*sln);
     P(r,:) = events_probabilities(segment);
end
```

Listing A15

Code scripted to obtain the distribution of the probability of the occurrence of the 9 types of Events for 3000 sound tracks of 3 seconds duration each.

```
% Probability of 9 different events and 2 categories; the stream being
% split into tracks. Histogram of 3000 frames to show validity of method

%% Declare constants and empty matrices
clear all
             fs = 44100; % Sampling Frequency
              r = 60;    % Accumulate 60 seconds
  target_tracks = 3000;  % Target number of tracks to be analyzed
analyzed_tracks = 0;     % Analyzed number of tracks
          split = 3;     % Each r seconds are split into split seconds

Events_speech = [];  % Empty matrix to hold probabilities of events for
speech
   Events_pop = [];  % Empty matrix to hold probabilities of events for
music

speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword'; % Folder containing
speech data
   pop_folder = 'C:\DATA\MUSIC\pop_music\popmusic';   % Folder containing
music data

speech_counter = 0;     % Initialize Speech Files Counter
   pop_counter = 0;     % Initialize Music Files Counter

%% Compute probabilities of events for each type of audio

while analyzed_tracks < target_tracks  % Repeat analysis until
target_tracks reached

% Analyze speech signals

duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
     speech = wavread([speech_folder,num2str(speech_counter)]);
     speech = speech(:,1);
     speech = remove_silent_frames(speech, fs, 500e-3, 0.1);
    speech_ = [speech_; speech];
   duration = length(speech_);
end

   speech = speech_(1:r*fs);
   speech = speech-mean(speech);
   speech = speech/max(abs(speech));
Ps_speech = events_probabilities_per_track(speech,fs,split);
```

Listing A15 (Continued ...)

```
% Analyze pop music signals

duration = 0;
    pop_ = [];
while duration < r*fs
if pop_counter == 4094
        pop_counter = 0;
end
pop_counter = pop_counter + 1;
    pop = wavread([pop_folder,num2str(pop_counter)]);
    pop = pop(:,1);
    pop = remove_silent_frames(pop, fs, 500e-3, 0.1);
   pop_ = [pop_; pop];
 duration = length(pop_);
end
    pop = pop_(1:fix(r*fs));
    pop = pop-mean(pop);
    pop = pop/max(abs(pop));
 Ps_pop = events_probabilities_per_track(pop,fs,split);

% Store each type of audio signal in its matrix

  Events_speech = [Events_speech; Ps_speech];
     Events_pop = [Events_pop;       Ps_pop];

% Compute the number of tracks analyzed so far for each type of audio

[Rows1,Columns1] = size(Events_speech);
[Rows2,Columns2] = size(Events_pop);

analyzed_tracks = min([Rows1, Rows2]);

[num2str(analyzed_tracks),' tracks completed .. ']

end

Events_speech = Events_speech(1:target_tracks,:);
   Events_pop = Events_pop(1:target_tracks,:);

save speech_pop
```

Listing A15 (Continued ...)

```
%% ------------------- plot P(NZCE) (event + +) --------------------

Bins = 0:0.0125:1; % Bins used for histograms

event = 6;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;

figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0 1])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(NZCE_+ _+); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech_6,TP_speech_6,Distance_speech_6, ...
                    opt_TP_speech_6,opt_FP_speech_6,opt_thr_speech_6] = ...
                                        roc(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_6,TP_speech_6, 'b', 'LineWidth', 3)

text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_6))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_6))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_6))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_6,TP_speech_6)))

% Detect music in a speech/music database

[FP_music_6,TP_music_6,Distance_music_6, ...
                    opt_TP_music_6,opt_FP_music_6,opt_thr_music_6] = ...
                                        roc2(Events_pop_Y, Events_speech_Y);
```

Listing A15 (Continued ...)

```
plot(FP_music_6,TP_music_6, 'r', 'LineWidth', 3)
title('ROC Curves Using P(NZCE_+ _+)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')

text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_6))
text(0.8,0.3, num2str(opt_FP_music_6))
text(0.8,0.2, num2str(opt_thr_music_6))
text(0.8,0.1, num2str(trapz(FP_music_6,TP_music_6)))

axis([-0.01 1 0 1.01])

%% -------------------- plot P(NZCE) (event - -) --------------------

event = 7;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;

figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0 1])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(NZCE_- _-); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech_7,TP_speech_7,Distance_speech_7, ...
                    opt_TP_speech_7,opt_FP_speech_7,opt_thr_speech_7] = ...
                                       roc2(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_7,TP_speech_7, 'b', 'LineWidth', 3)
```

Listing A15 (Continued ...)

```
text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_7))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_7))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_7))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_7,TP_speech_7)))

% Detect music in a speech/music database
[FP_music_7,TP_music_7,Distance_music_7, ...
                    opt_TP_music_7,opt_FP_music_7,opt_thr_music_7] = ...
                                          roc(Events_pop_Y, Events_speech_Y);

plot(FP_music_7,TP_music_7, 'r', 'LineWidth', 3)
title('ROC Curves Using P(NZCE_- _-)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')

text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_7))
text(0.8,0.3, num2str(opt_FP_music_7))
text(0.8,0.2, num2str(opt_thr_music_7))
text(0.8,0.1, num2str(trapz(FP_music_7,TP_music_7)))

axis([-0.01 1 0 1.01])

%% -------------------- plot P(ZCE) (event + -) --------------------

Bins = 0:0.00125:1; % Bins used for histograms

event = 8;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;

figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0 0.2])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(ZCE_+ _-); ' ,num2str(target_tracks),' tracks; ', ...
                                          num2str(split), ' seconds each']);
```

Listing A15 (Continued ...)

```
% Detect speech in a speech/music database

[FP_speech_8,TP_speech_8,Distance_speech_8, ...
                    opt_TP_speech_8,opt_FP_speech_8,opt_thr_speech_8] = ...
                                        roc(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_8,TP_speech_8, 'b', 'LineWidth', 3)

text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_8))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_8))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_8))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_8,TP_speech_8)))

% Detect music in a speech/music database

[FP_music_8,TP_music_8,Distance_music_8, ...
                    opt_TP_music_8,opt_FP_music_8,opt_thr_music_8] = ...
                                        roc2(Events_pop_Y, Events_speech_Y);

plot(FP_music_8,TP_music_8, 'r', 'LineWidth', 3)
title('ROC Curves Using P(ZCE_+ _-)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')

text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_8))
text(0.8,0.3, num2str(opt_FP_music_8))
text(0.8,0.2, num2str(opt_thr_music_8))
text(0.8,0.1, num2str(trapz(FP_music_8,TP_music_8)))

axis([-0.01 1 0 1.01])

%% -------------------- plot P(ZCE) (event - +) --------------------

event = 9;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;
```

Listing A15 (Continued ...)

```
figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0 0.2])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(ZCE_- _+); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech_9,TP_speech_9,Distance_speech_9, ...
                    opt_TP_speech_9,opt_FP_speech_9,opt_thr_speech_9] = ...
                                        roc(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_9,TP_speech_9, 'b', 'LineWidth', 3)

text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_9))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_9))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_9))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_9,TP_speech_9)))

% Detect music in a speech/music database

[FP_music_9,TP_music_9,Distance_music_9, ...
                    opt_TP_music_9,opt_FP_music_9,opt_thr_music_9] = ...
                                        roc2(Events_pop_Y, Events_speech_Y);

plot(FP_music_9,TP_music_9, 'r', 'LineWidth', 3)
title('ROC Curves Using P(ZCE_- _+)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')

text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_9))
text(0.8,0.3, num2str(opt_FP_music_9))
text(0.8,0.2, num2str(opt_thr_music_9))
text(0.8,0.1, num2str(trapz(FP_music_9,TP_music_9)))

axis([-0.01 1 0 1.01])
```

Listing A15 (Continued ...)

```
%% ----------------- plot NZCE set (0 0 or + + or - -) -----------------

event = 10;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;

figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0.7 1])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(NZCE set); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech_10,TP_speech_10,Distance_speech_10, ...
                 opt_TP_speech_10,opt_FP_speech_10,opt_thr_speech_10] = ...
                                       roc2(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_10,TP_speech_10, 'b', 'LineWidth', 3)

text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_10))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_10))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_10))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_10,TP_speech_10)))

% Detect music in a speech/music database

[FP_music_10,TP_music_10,Distance_music_10, ...
                   opt_TP_music_10,opt_FP_music_10,opt_thr_music_10] = ...
                                       roc(Events_pop_Y, Events_speech_Y);

plot(FP_music_10,TP_music_10, 'r', 'LineWidth', 3)
title('ROC Curves Using P(NZCE set)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')
```

Listing A15 (Continued ...)

```
text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_10))
text(0.8,0.3, num2str(opt_FP_music_10))
text(0.8,0.2, num2str(opt_thr_music_10))
text(0.8,0.1, num2str(trapz(FP_music_10,TP_music_10)))

axis([-0.01 1 0 1.01])

%% -------------------- plot ZCE set (+ - or - +) --------------------

event = 11;

[Events_speech_Y, Events_speech_X] = rhist(Events_speech(:,event), Bins);
[Events_pop_Y,       Events_pop_X] = rhist(Events_pop(:,event),    Bins);

Events_speech_Y = 100 * Events_speech_Y;
   Events_pop_Y = 100 * Events_pop_Y;

figure('color','white'), hold on

plot(Events_speech_X, Events_speech_Y,  'b', 'LineWidth', 3)
plot(Events_pop_X,    Events_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')

xlim([0 0.3])
xlabel('Probability of Event')
ylabel('Percentage of tracks, %')
title(['P(ZCE set); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech_11,TP_speech_11,Distance_speech_11, ...
                 opt_TP_speech_11,opt_FP_speech_11,opt_thr_speech_11] = ...
                                        roc(Events_speech_Y, Events_pop_Y);

figure('color','white'), hold on
plot(FP_speech_11,TP_speech_11, 'b', 'LineWidth', 3)

text(0.5, 0.5,'speech')
text(0.05,0.4,'Optimum TP rate')
text(0.5, 0.4,num2str(opt_TP_speech_11))
text(0.05,0.3,'Optimum FP rate')
text(0.5, 0.3,num2str(opt_FP_speech_11))
text(0.05,0.2,'Optimum Threshold')
text(0.5, 0.2,num2str(opt_thr_speech_11))
text(0.05,0.1,'Area Under Curve')
text(0.5, 0.1,num2str(trapz(FP_speech_11,TP_speech_11)))
```

Listing A15 (Continued ...)

```
% Detect music in a speech/music database

[FP_music_11,TP_music_11,Distance_music_11, ...
                    opt_TP_music_11,opt_FP_music_11,opt_thr_music_11] = ...
                                        roc2(Events_pop_Y, Events_speech_Y);
plot(FP_music_11,TP_music_11, 'r', 'LineWidth', 3)
title('ROC Curves Using P(ZCE set)')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('Speech','Pop Music')
text(0.8,0.5,'Music')
text(0.8,0.4, num2str(opt_TP_music_11))
text(0.8,0.3, num2str(opt_FP_music_11))
text(0.8,0.2, num2str(opt_thr_music_11))
text(0.8,0.1, num2str(trapz(FP_music_11,TP_music_11)))
axis([-0.01 1 0 1.01])

%% -------------------- Compare all ROC curves --------------------
figure('color','white'), hold on
plot(FP_speech_6, TP_speech_6,  'b', 'LineWidth', 3)
plot(FP_speech_7, TP_speech_7,  'r', 'LineWidth', 3)
plot(FP_speech_8, TP_speech_8,  'k', 'LineWidth', 3)
plot(FP_speech_9, TP_speech_9,  'k', 'LineWidth', 3)
plot(FP_speech_10,TP_speech_10, 'k', 'LineWidth', 3)
plot(FP_speech_11,TP_speech_11, 'k', 'LineWidth', 3)
title('ROC Curves for Speech Detection')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('P(NZCE_+ _+)','P(NZCE_- _-)',...
'P(ZCE_+ _-)','P(ZCE_- _+)',...
'P(NZCE set)','P(ZCE set)')
axis([-0.01 1 0 1.01])

%% -------------------- Summary of parameters --------------------

'Speech detection'
[opt_TP_speech_6, opt_FP_music_6, opt_thr_speech_6, Distance_speech_6,
trapz(FP_speech_6, TP_speech_6 )
 opt_TP_speech_7, opt_FP_music_7, opt_thr_speech_7, Distance_speech_7,
trapz(FP_speech_7, TP_speech_7 )
 opt_TP_speech_8, opt_FP_music_8, opt_thr_speech_8, Distance_speech_8,
trapz(FP_speech_8, TP_speech_8 )
 opt_TP_speech_9, opt_FP_music_9, opt_thr_speech_9, Distance_speech_9,
trapz(FP_speech_9, TP_speech_9 )
opt_TP_speech_10,opt_FP_music_10,opt_thr_speech_10,Distance_speech_10,trapz
(FP_speech_10,TP_speech_10)
opt_TP_speech_11,opt_FP_music_11,opt_thr_speech_11,Distance_speech_11,trapz
(FP_speech_11,TP_speech_11)]

'Music detection'
```

Listing A15 (Continued ...)

```
 [opt_TP_music_6, opt_FP_music_6, opt_thr_music_6, Distance_music_6,
trapz(FP_music_6, TP_music_6 )
 opt_TP_music_7, opt_FP_music_7, opt_thr_music_7, Distance_music_7,
trapz(FP_music_7, TP_music_7 )
 opt_TP_music_8, opt_FP_music_8, opt_thr_music_8, Distance_music_8,
trapz(FP_music_8, TP_music_8 )
 opt_TP_music_9, opt_FP_music_9, opt_thr_music_9, Distance_music_9,
trapz(FP_music_9, TP_music_9 )

opt_TP_music_10,opt_FP_music_10,opt_thr_music_10,Distance_music_10,trapz(FP
_music_10,TP_music_10)

opt_TP_music_11,opt_FP_music_11,opt_thr_music_11,Distance_music_11,trapz(FP
_music_11,TP_music_11)]

%% ------------------- Pairs of features --------------------

figure('color','white'), hold on
plot(Events_speech(:,6), Events_speech(:,7), 'b.', 'LineWidth', 3)
plot(Events_pop(:,6),    Events_pop(:,7),    'r.', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlabel('P(NZCE_+ _+)')
ylabel('P(NZCE_- _-)')
title(['P(NZCE_- _-) against P(NZCE_+ _+) for ' , ...
     num2str(target_tracks),' tracks; ', num2str(split), ' seconds each']);
axis([0.2 1 0 0.8])

figure('color','white'), hold on
plot(Events_speech(:,6), Events_speech(:,8), 'b.', 'LineWidth', 3)
plot(Events_pop(:,6),    Events_pop(:,8),    'r.', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlabel('P(NZCE_+ _+)')
ylabel('P(ZCE_+ _-)')
title(['P(ZCE_+ _-) against P(NZCE_+ _+) for ' , ...
     num2str(target_tracks),' tracks; ', num2str(split), ' seconds each']);
axis([0 1 0 0.2])


figure('color','white'), hold on

plot(Events_speech(:,7), Events_speech(:,8), 'b.', 'LineWidth', 3)
plot(Events_pop(:,7),    Events_pop(:,8),    'r.', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlabel('P(NZCE_- _-)')
ylabel('P(ZCE_+ _-)')
title(['P(ZCE_+ _-) againts P(NZCE_- _-) for ' , ...
     num2str(target_tracks),' tracks; ', num2str(split), ' seconds each']);
axis([0 1 0 0.2])
```

Listing A16

Code scripted as a function to be used for computing the probability of the variation in positive and negative NZCE, known as δNZCE

```
% This function computes the probability of the variation in NZCE between
% positive and negative types of NZCE.

function Probability = delta_P_NZCE(input)

N = length(input);

positive = zeros(1,N);
negative = zeros(1,N);

positive(find(input>0)) = 1;
negative(find(input<0)) = 1;

count_positive = sum(positive(2:N).*positive(1:N-1));
count_negative = sum(negative(2:N).*negative(1:N-1));

Probability = abs(count_positive-count_negative)/(N-1);
```

Listing A17

Code scripted for computing the probability of the variation in positive and negative NZCE, known as δNZCE, for each track when splitting a stream of audio into short tracks, e.g. 3 seconds each

```
% This function splits the input signal into tracks of "split" seconds each
% and computes the probability of delta NZCE to occur in each track.

function Probability = delta_nzce_per_track(signal, fs, split)

       size = length(signal);              % length of the input signal

        sln = floor(split * fs);           % segment length = split seconds

   segments = floor(size/sln);             % Number of segments

Probability = zeros(segments,1);            % Reserve a zero vector

for r = 1 : segments
    segment = signal((r-1)*sln+1:r*sln);
Probability(r,1) = delta_P_NZCE(segment);
end
```

Listing A18

Code scripted for examining the effect of varying the track length on the probability of the positive and negative NZCE

```
% Analysis of the effect of changing the track length on the NZCE++/NZCE--

%% Declare constants and empty matrices
clear all
             fs = 44100; % Sampling Frequency
              r = 60;    % Accumulate 60 seconds
  target_tracks = 3000;  % Target number of tracks to be analyzed

range = [0.25, 0.5, 0.75, 1.0, 1.25, 2.5, 5.0, 10]; %range of track lengths

  Pos_PNZCE_speech_track = [];
     Pos_PNZCE_pop_track = [];
  Neg_PNZCE_speech_track = [];
     Neg_PNZCE_pop_track = [];

speech_folder='C:\DATA\SPEECH\UCL\WORDS\am\amword'; %Folder containing
speech
   pop_folder='C:\DATA\MUSIC\pop_music\popmusic';   %Folder containing
music

speech_counter = 0;     % Initialize Speech Files Counter
   pop_counter = 0;     % Initialize Music Files Counter

%% Compute probabilities of events for each type of audio

for track_length = 1 : length(range);
Pos_PNZCE_speech = [];
Neg_PNZCE_speech = [];
   Pos_PNZCE_pop = [];
   Neg_PNZCE_pop = [];

split = range(track_length);

analyzed_tracks = 0;                    % Analyzed number of tracks

while analyzed_tracks < target_tracks  % Repeat analysis until
target_tracks

% Analyze speech signals

duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
speech =
wavread([speech_folder,num2str(speech_counter)]);speech=speech(:,1);
    speech_ = [speech_; speech]; duration = length(speech_);
end
```

Listing A18 (Continued ...)

```
   speech = speech_(1:r*fs);
   speech = speech-mean(speech); speech = speech/max(abs(speech));

[PNZCE_pos_speech, PNZCE_neg_speech] = pnzce_per_track(speech,fs,split);

% Analyze pop music signals

duration = 0;
    pop_ = [];
while duration < r*fs
if pop_counter == 4094
        pop_counter = 0;
end
pop_counter = pop_counter + 1;
    pop = wavread([pop_folder,num2str(pop_counter)]);
    pop = pop(:,1);
   pop_ = [pop_; pop];
 duration = length(pop_);
end
    pop = pop_(1:fix(r*fs)); pop = pop-mean(pop); pop = pop/max(abs(pop));

[PNZCE_pos_pop, PNZCE_neg_pop] = pnzce_per_track(pop,fs,split);

% Store each type of audio signal in its matrix

  Pos_PNZCE_speech = [Pos_PNZCE_speech; PNZCE_pos_speech];
  Neg_PNZCE_speech = [Neg_PNZCE_speech; PNZCE_neg_speech];
     Pos_PNZCE_pop = [Pos_PNZCE_pop;    PNZCE_pos_pop];
     Neg_PNZCE_pop = [Neg_PNZCE_pop;    PNZCE_neg_pop];

% Compute the number of tracks analyzed so far for each type of audio

[Rows1,Columns1] = size(Pos_PNZCE_speech);
[Rows2,Columns2] = size(Pos_PNZCE_pop);
 analyzed_tracks = min([Rows1, Rows2]);

[num2str(analyzed_tracks),' tracks completed .. each of length = ', ...
                                                  num2str(split), ' seconds']
end

% Update the main matrix holding results of all tracks

Pos_PNZCE_speech_track = [Pos_PNZCE_speech_track; ...
                                    Pos_PNZCE_speech(1:target_tracks)'];
   Pos_PNZCE_pop_track = [Pos_PNZCE_pop_track; ...
                                    Pos_PNZCE_pop(1:target_tracks)'   ];
Neg_PNZCE_speech_track = [Neg_PNZCE_speech_track; ...
                                    Neg_PNZCE_speech(1:target_tracks)'];
   Neg_PNZCE_pop_track = [Neg_PNZCE_pop_track; ...
                                    Neg_PNZCE_pop(1:target_tracks)'   ];
end

save speech_pop_track_length
```

Listing A18 (Continued ...)

```
%% ------------------- plot P(NZCE) (event + +) --------------------

Bins = 0:0.0125:1; % Bins used for histograms

  opt_TP_speech_track = [];
  opt_FP_speech_track = [];
 opt_thr_speech_track = [];
Distance_speech_track = [];
ROC_area_speech_track = [];

  opt_TP_music_track = [];
  opt_FP_music_track = [];
 opt_thr_music_track = [];
Distance_music_track = [];
ROC_area_music_track = [];

figure('color','white'), hold on
figure('color','white'), hold on

for track_length = 1 : length(range);

split = range(track_length);

[NZCE_speech_Y, NZCE_speech_X] =
rhist(Pos_PNZCE_speech_track(track_length,:)', Bins);
[NZCE_pop_Y,    NZCE_pop_X   ] =
rhist(Pos_PNZCE_pop_track(track_length,:)',    Bins);

NZCE_speech_Y = 100 * NZCE_speech_Y;
   NZCE_pop_Y = 100 * NZCE_pop_Y;

figure('color','white'), hold on
plot(NZCE_speech_X, NZCE_speech_Y,  'b', 'LineWidth', 3)
plot(NZCE_pop_X,    NZCE_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlim([0 1])
xlabel('P(NZCE_+ _+)')
ylabel('Percentage of tracks, %')
title(['P(NZCE_+ _+); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech,TP_speech,Distance_speech,opt_TP_speech,opt_FP_speech,...
                          opt_thr_speech] = roc(NZCE_speech_Y, NZCE_pop_Y);
  opt_TP_speech_track = [opt_TP_speech_track;  opt_TP_speech];
  opt_FP_speech_track = [opt_FP_speech_track;  opt_FP_speech];
 opt_thr_speech_track = [opt_thr_speech_track; opt_thr_speech];
Distance_speech_track = [Distance_speech_track;Distance_speech];
ROC_area_speech_track = [ROC_area_speech_track;trapz(FP_speech,TP_speech)];

figure(1), plot(FP_speech,TP_speech, 'b', 'LineWidth', 3)
```

Listing A18 (Continued ...)

```
% Detect music in a speech/music database

[FP_music,TP_music,Distance_music,opt_TP_music,opt_FP_music,...
                          opt_thr_music] = roc2(NZCE_pop_Y, NZCE_speech_Y);

  opt_TP_music_track = [opt_TP_music_track;   opt_TP_music];
  opt_FP_music_track = [opt_FP_music_track;   opt_FP_music];
 opt_thr_music_track = [opt_thr_music_track;  opt_thr_music];
Distance_music_track = [Distance_music_track; Distance_music];
ROC_area_music_track = [ROC_area_music_track; trapz(FP_music,TP_music)];

figure(2), plot(FP_music,TP_music, 'r', 'LineWidth', 3)

end

figure(1)
title('ROC Curves Using P(NZCE_+ _+) to detect speech')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('0.25 second','0.5 second', '0.75 second', '1 second', ...
'1.25 second', '2.5 second', '5 second', '10 second')

axis([-0.01 1 0 1.01])

figure(2)
title('ROC Curves Using P(NZCE_+ _+) to detect music')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('0.25 second','0.5 second', '0.75 second', '1 second', ...
'1.25 second', '2.5 second', '5 second', '10 second')

axis([-0.01 1 0 1.01])

figure('color','white'), grid, hold on
plot(range, opt_TP_speech_track, 'b.-', 'LineWidth', 3)
plot(range, opt_TP_music_track,  'r.-', 'LineWidth', 3)
plot(range, opt_FP_speech_track, 'b--', 'LineWidth', 3)
plot(range, opt_FP_music_track,  'r--', 'LineWidth', 3)
title('Effect of Increasing Track Length on TP/FP rates for P(NZCE_+ _+)')
xlabel('Track Length, seconds')
ylabel('TP/FP Rates')
legend('Speech TP Rate', 'Music TP Rate', ...
'Speech FP Rate', 'Music FP Rate')

figure('color','white'), grid, hold on
plot(range, Distance_speech_track, 'b', 'LineWidth', 3)
plot(range, Distance_music_track,  'r', 'LineWidth', 3)
title('Effect of Track Length on Euclidian Distance for P(NZCE_+ _+)')
xlabel('Track Length, seconds')
ylabel('Euclidian Distance')
legend('Speech', 'Music')
```

Listing A18 (Continued ...)

```
%% ------------------- plot P(NZCE) (event - -) --------------------

Bins = 0:0.0125:1; % Bins used for histograms

  opt_TP_speech_track = [];
  opt_FP_speech_track = [];
 opt_thr_speech_track = [];
Distance_speech_track = [];
ROC_area_speech_track = [];

  opt_TP_music_track = [];
  opt_FP_music_track = [];
 opt_thr_music_track = [];
Distance_music_track = [];
ROC_area_music_track = [];

figure('color','white'), hold on
figure('color','white'), hold on

for track_length = 1 : length(range);

split = range(track_length);

[NZCE_speech_Y, NZCE_speech_X] =
rhist(Neg_PNZCE_speech_track(track_length,:)', Bins);
[NZCE_pop_Y,    NZCE_pop_X   ] =
rhist(Neg_PNZCE_pop_track(track_length,:)',    Bins);

NZCE_speech_Y = 100 * NZCE_speech_Y;
   NZCE_pop_Y = 100 * NZCE_pop_Y;

figure('color','white'), hold on
plot(NZCE_speech_X, NZCE_speech_Y,  'b', 'LineWidth', 3)
plot(NZCE_pop_X,    NZCE_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlim([0 1])
xlabel('P(NZCE_- _-)')
ylabel('Percentage of tracks, %')
title(['P(NZCE_- _-); ' ,num2str(target_tracks),' tracks; ', ...
                                        num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech,TP_speech,Distance_speech,opt_TP_speech,opt_FP_speech,...
                         opt_thr_speech] = roc2(NZCE_speech_Y, NZCE_pop_Y);


  opt_TP_speech_track = [opt_TP_speech_track;  opt_TP_speech];
  opt_FP_speech_track = [opt_FP_speech_track;  opt_FP_speech];
 opt_thr_speech_track = [opt_thr_speech_track; opt_thr_speech];
Distance_speech_track = [Distance_speech_track;Distance_speech];
ROC_area_speech_track = [ROC_area_speech_track;trapz(FP_speech,TP_speech)];

figure(1)
plot(FP_speech,TP_speech, 'b', 'LineWidth', 3)
```

Listing A18 (Continued ...)

```
% Detect music in a speech/music database

[FP_music,TP_music,Distance_music,opt_TP_music,opt_FP_music,...
                            opt_thr_music] = roc(NZCE_pop_Y, NZCE_speech_Y);

  opt_TP_music_track = [opt_TP_music_track;   opt_TP_music];
  opt_FP_music_track = [opt_FP_music_track;   opt_FP_music];
 opt_thr_music_track = [opt_thr_music_track;  opt_thr_music];
Distance_music_track = [Distance_music_track; Distance_music];
ROC_area_music_track = [ROC_area_music_track; trapz(FP_music,TP_music)];

figure(2)
plot(FP_music,TP_music, 'r', 'LineWidth', 3)

end

figure(1)
title('ROC Curves Using P(NZCE_- _-) to detect speech')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('0.25 second','0.5 second', '0.75 second', '1 second', ...
'1.25 second', '2.5 second', '5 second', '10 second')

axis([-0.01 1 0 1.01])

figure(2)
title('ROC Curves Using P(NZCE_- _-) to detect music')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('0.25 second','0.5 second', '0.75 second', '1 second', ...
'1.25 second', '2.5 second', '5 second', '10 second')

axis([-0.01 1 0 1.01])

figure('color','white'), grid, hold on
plot(range, opt_TP_speech_track, 'b.-', 'LineWidth', 3)
plot(range, opt_TP_music_track,  'r.-', 'LineWidth', 3)
plot(range, opt_FP_speech_track, 'b--', 'LineWidth', 3)
plot(range, opt_FP_music_track,  'r--', 'LineWidth', 3)
title('Effect of Track Length on TP/FP rates for P(NZCE_- _-)')
xlabel('Track Length, seconds')
ylabel('TP/FP Rates')
legend('Speech TP Rate', 'Music TP Rate', ...
'Speech FP Rate', 'Music FP Rate')


figure('color','white'), grid, hold on
plot(range, Distance_speech_track, 'b', 'LineWidth', 3)
plot(range, Distance_music_track,  'r', 'LineWidth', 3)
title('Effect of Track Length on Euclidian Distance for P(NZCE_- _-)')
xlabel('Track Length, seconds')
ylabel('Euclidian Distance')
legend('Speech', 'Music')
```

Listing A19

Code scripted to examine the effect of room noise on the discrimination process using either types of NZCE.

```
% Probability of 9 different events and 2 categories; tracks being
% split into frames. Histogram of 3000 frames to show validity of method

%% Declare constants and empty matrices
clear all
               fs = 44100;    % Sampling Frequency
                r = 60;       % Accumulate 300 seconds ( 5 minutes ) to from
% a single audio track
    target_tracks = 3000;     % Target number of tracks to be analyzed
            split = 3;        % Each r seconds are split into split seconds

P_NZCE_pos_Noisy_speech = []; % P(NZCE++) for noisy speech
P_NZCE_neg_Noisy_speech = []; % P(NZCE--) for noisy speech
 P_NZCE_pos_Noisy_music = []; % P(NZCE++) for noisy music
 P_NZCE_neg_Noisy_music = []; % P(NZCE--) for noisy music

SNR_speech = []; % Signal to Noise Ratio using RMS Power for speech
 SNR_music = []; % Signal to Noise Ratio using RMS Power for music

    speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword';
       pop_folder = 'C:\DATA\MUSIC\pop_music\popmusic';

   speech_counter = 0;
      pop_counter = 0;

%% Record 60 seconds of room noise

% room_noise = wavrecord(60*fs,fs);
% room_noise = room_noise - mean(room_noise);
% wavwrite(room_noise, fs, 'room_noise.wav')

room_noise = wavread('room_noise');
      time = (1:length(room_noise))/fs;

%% Display Probability Density Function of the Room Noise


[y,x] = rhist(room_noise,100);

figure('color','white')
plot(time,room_noise);
xlabel('Time, seconds')
ylabel('Amplitude, V')
title('Room noise as measured from a commercial condensor microphone')
axis([0 5 -0.01 0.01])
```

Listing A19 (Continued ...)

```
figure('color','white')
plot(x,y, 'LineWidth', 3)
xlabel('Amplitude, V')
ylabel('Probability Density')
title('Probability Density Function of Room Noise')
xlim([-.01 0.01])

%% Compute probabilities of events for each type of audio

noise = room_noise;

for noise_gain = 0.1 : 0.3 : 3

P_NZCE_pos_speech = [];
P_NZCE_neg_speech = [];
 P_NZCE_pos_music = [];
 P_NZCE_neg_music = [];

  analyzed_tracks = 0;  % Analyzed number of tracks

while analyzed_tracks < target_tracks

% Analyze speech signals

duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
         speech = wavread([speech_folder,num2str(speech_counter)]);
         speech = speech(:,1);
        speech_ = [speech_; speech];
       duration = length(speech_);
end

      speech = speech_(1:r*fs);
      speech = speech-mean(speech);
      speech = speech/max(abs(speech));
Noisy_speech = speech + noise_gain*noise;
[P_NZCE_pos_speech_, P_NZCE_neg_speech_] = ...
                                      pnzce_per_track(Noisy_speech,fs,split);
```

Listing A19 (Continued ...)

```
% Analyze pop music signals

duration = 0;
    pop_ = [];
while duration < r*fs
if pop_counter == 4094
        pop_counter = 0;
end
pop_counter = pop_counter + 1;
        pop = wavread([pop_folder,num2str(pop_counter)]);
        pop = pop(:,1);
       pop_ = [pop_; pop];
   duration = length(pop_);
end
        pop = pop_(1:fix(r*fs));
        pop = pop-mean(pop);
        pop = pop/max(abs(pop));
  Noisy_pop = pop + noise_gain*noise;
[P_NZCE_pos_music_, P_NZCE_neg_music_] = ...
                                        pnzce_per_track(Noisy_pop, fs, split);

% Compute the number of tracks analyzed so far for each type of audio

P_NZCE_pos_speech = [P_NZCE_pos_speech; P_NZCE_pos_speech_];
P_NZCE_neg_speech = [P_NZCE_neg_speech; P_NZCE_neg_speech_];
 P_NZCE_pos_music = [P_NZCE_pos_music;  P_NZCE_pos_music_];
 P_NZCE_neg_music = [P_NZCE_neg_music;  P_NZCE_neg_music_];

[A1, B1] = size(P_NZCE_pos_speech);

[A2, B2] = size(P_NZCE_pos_music);

analyzed_tracks = min([A1, A2]);

[num2str(analyzed_tracks),' tracks analyzed']

end


SNR_speech_ = 20*log10(rms(speech)/rms(noise_gain*noise));

 SNR_music_ = 20*log10(rms(pop)/rms(noise_gain*noise));

[' SNR for speech = ', num2str(SNR_speech_), ' dB']

[' SNR for music  = ', num2str(SNR_speech_), ' dB']


SNR_speech = [SNR_speech; SNR_speech_];

 SNR_music = [SNR_music;  SNR_music_];
```

Listing A19 (Continued ...)

```
P_NZCE_pos_Noisy_speech = [P_NZCE_pos_Noisy_speech; ...
                                        P_NZCE_pos_speech(1:target_tracks)'];
P_NZCE_neg_Noisy_speech = [P_NZCE_neg_Noisy_speech; ...
                                        P_NZCE_neg_speech(1:target_tracks)'];
 P_NZCE_pos_Noisy_music = [P_NZCE_pos_Noisy_music;  ...
                                         P_NZCE_pos_music(1:target_tracks)'];
 P_NZCE_neg_Noisy_music = [P_NZCE_neg_Noisy_music;  ...
                                         P_NZCE_neg_music(1:target_tracks)'];
end


save Noisy_Speech_Noisy_music

%% -------------------- plot P(NZCE) (event + +) --------------------

Bins = 0:0.0125:1; % Bins used for histograms

  opt_TP_Noisy_speech_pos = [];
  opt_FP_Noisy_speech_pos = [];
 opt_thr_Noisy_speech_pos = [];
Distance_Noisy_speech_pos = [];
ROC_area_Noisy_speech_pos = [];

   opt_TP_Noisy_music_pos = [];
   opt_FP_Noisy_music_pos = [];
  opt_thr_Noisy_music_pos = [];
 Distance_Noisy_music_pos = [];
 ROC_area_Noisy_music_pos = [];

figure('color','white'), hold on
figure('color','white'), hold on

for noise_case = 1: 10

[NZCE_speech_Y, NZCE_speech_X] =
rhist(P_NZCE_pos_Noisy_speech(noise_case,:)', Bins);
[NZCE_pop_Y,    NZCE_pop_X   ] =
rhist(P_NZCE_pos_Noisy_music(noise_case,:)',  Bins);

NZCE_speech_Y = 100 * NZCE_speech_Y;
   NZCE_pop_Y = 100 * NZCE_pop_Y;

figure('color','white'), hold on
plot(NZCE_speech_X, NZCE_speech_Y,  'b', 'LineWidth', 3)
plot(NZCE_pop_X,    NZCE_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlim([0 1])
xlabel('P(NZCE_+ _+)')
ylabel('Percentage of tracks, %')
title(['P(NZCE_+ _+); ' ,num2str(target_tracks),' tracks; ', 'SNR = ', ...
num2str(SNR_speech(noise_case)), 'dB; ', num2str(split), ' seconds each']);
```

Listing A19 (Continued ...)

```
% Detect speech in a speech/music database

[FP_speech,TP_speech,Distance_speech,opt_TP_speech,opt_FP_speech,...
                           opt_thr_speech] = roc(NZCE_speech_Y, NZCE_pop_Y);

ROC_Area_speech = trapz(FP_speech,TP_speech);

  opt_TP_Noisy_speech_pos = [opt_TP_Noisy_speech_pos;   opt_TP_speech];
  opt_FP_Noisy_speech_pos = [opt_FP_Noisy_speech_pos;   opt_FP_speech];
 opt_thr_Noisy_speech_pos = [opt_thr_Noisy_speech_pos;  opt_thr_speech];
Distance_Noisy_speech_pos = [Distance_Noisy_speech_pos; Distance_speech];
ROC_area_Noisy_speech_pos = [ROC_area_Noisy_speech_pos; ROC_Area_speech];

figure(1)
plot(FP_speech,TP_speech, 'b', 'LineWidth', 3)

% Detect music in a speech/music database

[FP_music,TP_music,Distance_music,opt_TP_music,opt_FP_music,...
                           opt_thr_music] = roc2(NZCE_pop_Y, NZCE_speech_Y);

ROC_Area_music = trapz(FP_music,TP_music);

  opt_TP_Noisy_music_pos = [opt_TP_Noisy_music_pos;   opt_TP_music];
  opt_FP_Noisy_music_pos = [opt_FP_Noisy_music_pos;   opt_FP_music];
 opt_thr_Noisy_music_pos = [opt_thr_Noisy_music_pos;  opt_thr_music];
Distance_Noisy_music_pos = [Distance_Noisy_music_pos; Distance_music];
ROC_area_Noisy_music_pos = [ROC_area_Noisy_music_pos; ROC_Area_music];

figure(2)
plot(FP_music,TP_music, 'r', 'LineWidth', 3)

end

figure(1)
title('ROC Curves Using P(NZCE_+ _+) to detect speech')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend(['SNR = ', num2str(SNR_speech(1)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(2)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(3)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(4)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(5)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(6)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(7)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(8)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(9)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(10)), ' dB'] )

axis([-0.01 1 0 1.01])
```

Listing A19 (Continued ...)

```
figure(2)
title('ROC Curves Using P(NZCE_+ _+) to detect music')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend(['SNR = ', num2str(SNR_music(1)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(2)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(3)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(4)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(5)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(6)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(7)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(8)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(9)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(10)), ' dB'] )

axis([-0.01 1 0 1.01])

figure('color','white'), grid, hold on
plot(SNR_speech, opt_TP_Noisy_speech_pos, 'b.-', 'LineWidth', 3)
plot(SNR_music,  opt_TP_Noisy_music_pos,  'r.-', 'LineWidth', 3)
plot(SNR_speech, opt_FP_Noisy_speech_pos, 'b--', 'LineWidth', 3)
plot(SNR_music,  opt_FP_Noisy_music_pos,  'r--', 'LineWidth', 3)
title('Effect of SNR on the TP/FP rates for P(NZCE_+ _+)')
xlabel('SNR, dB')
ylabel('TP/FP Rates')
legend('Speech TP Rate', 'Music TP Rate', ...
'Speech FP Rate', 'Music FP Rate')

figure('color','white'), grid, hold on
plot(SNR_speech, Distance_Noisy_speech_pos, 'b', 'LineWidth', 3)
plot(SNR_music, Distance_Noisy_music_pos,   'r', 'LineWidth', 3)
title('Effect of SNR on the Euclidian Distance for P(NZCE_+ _+)')
xlabel('SNR, dB')
ylabel('Euclidian Distance')
legend('Speech', 'Music')

%% -------------------- plot P(NZCE) (event - -) --------------------

Bins = 0:0.0125:1; % Bins used for histograms

  opt_TP_Noisy_speech_neg = [];
  opt_FP_Noisy_speech_neg = [];
 opt_thr_Noisy_speech_neg = [];
Distance_Noisy_speech_neg = [];
ROC_area_Noisy_speech_neg = [];

   opt_TP_Noisy_music_neg = [];
   opt_FP_Noisy_music_neg = [];
  opt_thr_Noisy_music_neg = [];
 Distance_Noisy_music_neg = [];
 ROC_area_Noisy_music_neg = [];
```

Listing A19 (Continued ...)

```
figure('color','white'), hold on
figure('color','white'), hold on

for noise_case = 1: 10

[NZCE_speech_Y, NZCE_speech_X] =
rhist(P_NZCE_neg_Noisy_speech(noise_case,:)', Bins);
[NZCE_pop_Y,    NZCE_pop_X   ] =
rhist(P_NZCE_neg_Noisy_music(noise_case,:)',  Bins);

NZCE_speech_Y = 100 * NZCE_speech_Y;
   NZCE_pop_Y = 100 * NZCE_pop_Y;

figure('color','white'), hold on
plot(NZCE_speech_X, NZCE_speech_Y,  'b', 'LineWidth', 3)
plot(NZCE_pop_X,    NZCE_pop_Y,     'r', 'LineWidth', 3)
legend('Speech', 'Pop Music')
xlim([0 1])
xlabel('P(NZCE_- _-)')
ylabel('Percentage of tracks, %')
title(['P(NZCE_- _-); ' ,num2str(target_tracks),' tracks; ', 'SNR = ', ...
num2str(SNR_speech(noise_case)), 'dB; ', num2str(split), ' seconds each']);

% Detect speech in a speech/music database

[FP_speech,TP_speech,Distance_speech,opt_TP_speech,opt_FP_speech,...
                         opt_thr_speech] = roc2(NZCE_speech_Y, NZCE_pop_Y);

ROC_Area_speech = trapz(FP_speech,TP_speech);

  opt_TP_Noisy_speech_neg = [opt_TP_Noisy_speech_neg;   opt_TP_speech];
  opt_FP_Noisy_speech_neg = [opt_FP_Noisy_speech_neg;   opt_FP_speech];
 opt_thr_Noisy_speech_neg = [opt_thr_Noisy_speech_neg;  opt_thr_speech];
Distance_Noisy_speech_neg = [Distance_Noisy_speech_neg; Distance_speech];
ROC_area_Noisy_speech_neg = [ROC_area_Noisy_speech_neg; ROC_Area_speech];

figure(1)
plot(FP_speech,TP_speech, 'b', 'LineWidth', 3)

% Detect music in a speech/music database

[FP_music,TP_music,Distance_music,opt_TP_music,opt_FP_music,...
                           opt_thr_music] = roc(NZCE_pop_Y, NZCE_speech_Y);
ROC_Area_music = trapz(FP_music,TP_music);

  opt_TP_Noisy_music_neg = [opt_TP_Noisy_music_neg;   opt_TP_music];
  opt_FP_Noisy_music_neg = [opt_FP_Noisy_music_neg;   opt_FP_music];
 opt_thr_Noisy_music_neg = [opt_thr_Noisy_music_neg;  opt_thr_music];
Distance_Noisy_music_neg = [Distance_Noisy_music_neg; Distance_music];
ROC_area_Noisy_music_neg = [ROC_area_Noisy_music_neg; ROC_Area_music];
```

Listing A19 (Continued ...)

```
figure(2)
plot(FP_music,TP_music, 'r', 'LineWidth', 3)

end

figure(1)
title('ROC Curves Using P(NZCE_- _-) to detect speech')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend(['SNR = ', num2str(SNR_speech(1)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(2)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(3)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(4)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(5)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(6)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(7)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(8)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(9)),  ' dB'], ...
       ['SNR = ', num2str(SNR_speech(10)), ' dB'] )
axis([-0.01 1 0 1.01])

figure(2)
title('ROC Curves Using P(NZCE_- _-) to detect music')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend(['SNR = ', num2str(SNR_music(1)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(2)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(3)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(4)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(5)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(6)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(7)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(8)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(9)),  ' dB'], ...
       ['SNR = ', num2str(SNR_music(10)), ' dB'] )
axis([-0.01 1 0 1.01])

figure('color','white'), grid, hold on
plot(SNR_speech, opt_TP_Noisy_speech_neg, 'b.-', 'LineWidth', 3)
plot(SNR_music,  opt_TP_Noisy_music_neg,  'r.-', 'LineWidth', 3)
plot(SNR_speech, opt_FP_Noisy_speech_neg, 'b--', 'LineWidth', 3)
plot(SNR_music,  opt_FP_Noisy_music_neg,  'r--', 'LineWidth', 3)
title('Effect of SNR on the TP/FP rates for P(NZCE_- _-)')
xlabel('SNR, dB'); ylabel('TP/FP Rates')
legend('Speech TP Rate', 'Music TP Rate', ...
'Speech FP Rate', 'Music FP Rate')

figure('color','white'), grid, hold on
plot(SNR_speech, Distance_Noisy_speech_pos, 'b', 'LineWidth', 3)
plot(SNR_music, Distance_Noisy_music_pos,   'r', 'LineWidth', 3)
title('Effect of SNR on the Euclidian Distance for P(NZCE_- _-)')
xlabel('SNR, dB'); ylabel('Euclidian Distance'); legend('Speech', 'Music')
```

Listing A20

Code scripted to examine the effect of down-sampling the audio signal prior to computing the NZCE features.

```
% Examine the effect of downsampling the audio files to lower sampling
% rates.

%% Declare constants and empty matrices
clear all
               fs = 44100;    % Sampling Frequency
                r = 60;       % Accumulate 300 seconds ( 5 minutes ) to from
% a single audio track
    target_tracks = 3000;     % Target number of tracks to be analyzed
  analyzed_tracks = 0;        % Analyzed number of tracks
            split = 3;        % Each r seconds are split into split seconds

 PNZCE_pos_speech1 = []; PNZCE_neg_speech1 = [];
 PNZCE_pos_speech2 = []; PNZCE_neg_speech2 = [];
 PNZCE_pos_speech3 = []; PNZCE_neg_speech3 = [];
 PNZCE_pos_speech4 = []; PNZCE_neg_speech4 = [];
 PNZCE_pos_speech5 = []; PNZCE_neg_speech5 = [];
 PNZCE_pos_speech6 = []; PNZCE_neg_speech6 = [];
 PNZCE_pos_speech7 = []; PNZCE_neg_speech7 = [];

  PNZCE_pos_music1 = [];  PNZCE_neg_music1 = [];
  PNZCE_pos_music2 = [];  PNZCE_neg_music2 = [];
  PNZCE_pos_music3 = [];  PNZCE_neg_music3 = [];
  PNZCE_pos_music4 = [];  PNZCE_neg_music4 = [];
  PNZCE_pos_music5 = [];  PNZCE_neg_music5 = [];
  PNZCE_pos_music6 = [];  PNZCE_neg_music6 = [];
  PNZCE_pos_music7 = [];  PNZCE_neg_music7 = [];


    speech_folder = 'C:\DATA\SPEECH\UCL\WORDS\am\amword';
     music_folder = 'C:\DATA\MUSIC\pop_music\popmusic';

   speech_counter = 0;
    music_counter = 0;

    fs1 = fs;
    fs2 = 22050;     % Down-sampling to 22.05 kHz
    fs3 = 16000;     % Down-sampling to 16 kHz
    fs4 = 8000;      % Down-sampling to  8 kHz
    fs5 = 4000;      % Down-sampling to  4 kHz
    fs6 = 2000;      % Down-sampling to  2 kHz
    fs7 = 1000;      % Down-sampling to  1 kHz
```

Listing A20 (Continued ...)

```
%% Compute probabilities of events for each type of audio

while analyzed_tracks < target_tracks

% Analyze speech signals

duration = 0;
 speech_ = [];

while duration < r*fs
if speech_counter >= 1915
        speech_counter = 0;
end
speech_counter  = speech_counter + 1;
     speech = wavread([speech_folder,num2str(speech_counter)]);
     speech = speech(:,1);
    speech_ = [speech_; speech];
   duration = length(speech_);
end

   speech = speech_(1:r*fs);
   speech = speech-mean(speech);
  speech1 = speech/max(abs(speech));
  speech2 = resample(speech,fs2,fs1);  speech3 = resample(speech,fs3,fs1);
  speech4 = resample(speech,fs4,fs1);  speech5 = resample(speech,fs5,fs1);
  speech6 = resample(speech,fs6,fs1);  speech7 = resample(speech,fs7,fs1);

[PNZCE_pos_speech1_,PNZCE_neg_speech1_]=pnzce_per_track(speech1,fs1,split);
[PNZCE_pos_speech2_,PNZCE_neg_speech2_]=pnzce_per_track(speech2,fs2,split);
[PNZCE_pos_speech3_,PNZCE_neg_speech3_]=pnzce_per_track(speech3,fs3,split);
[PNZCE_pos_speech4_,PNZCE_neg_speech4_]=pnzce_per_track(speech4,fs4,split);
[PNZCE_pos_speech5_,PNZCE_neg_speech5_]=pnzce_per_track(speech5,fs5,split);
[PNZCE_pos_speech6_,PNZCE_neg_speech6_]=pnzce_per_track(speech6,fs6,split);
[PNZCE_pos_speech7_,PNZCE_neg_speech7_]=pnzce_per_track(speech7,fs7,split);

clear speech1speech2speech3speech4speech5speech6speech7

% Analyze pop music signals

duration = 0;
 music_ = [];

while duration < r*fs
if music_counter >= 4094
        music_counter = 0;
end
music_counter  = music_counter + 1;
     music = wavread([music_folder,num2str(music_counter)]);
     music = music(:,1);
    music_ = [music_; music];
   duration = length(music_);
end
```

Listing A20 (Continued ...)

```
   music = music_(1:r*fs);
   music = music-mean(music);
  music1 = music/max(abs(music));
  music2 = resample(music,fs2,fs1);  music3 = resample(music,fs3,fs1);
  music4 = resample(music,fs4,fs1);  music5 = resample(music,fs5,fs1);
  music6 = resample(music,fs6,fs1);  music7 = resample(music,fs7,fs1);

[PNZCE_pos_music1_,PNZCE_neg_music1_] = pnzce_per_track(music1,fs1,split);
[PNZCE_pos_music2_,PNZCE_neg_music2_] = pnzce_per_track(music2,fs2,split);
[PNZCE_pos_music3_,PNZCE_neg_music3_] = pnzce_per_track(music3,fs3,split);
[PNZCE_pos_music4_,PNZCE_neg_music4_] = pnzce_per_track(music4,fs4,split);
[PNZCE_pos_music5_,PNZCE_neg_music5_] = pnzce_per_track(music5,fs5,split);
[PNZCE_pos_music6_,PNZCE_neg_music6_] = pnzce_per_track(music6,fs6,split);
[PNZCE_pos_music7_,PNZCE_neg_music7_] = pnzce_per_track(music7,fs7,split);

clear music1music2music3music4music5music6music7

 PNZCE_pos_speech1 = [PNZCE_pos_speech1; PNZCE_pos_speech1_];
 PNZCE_neg_speech1 = [PNZCE_neg_speech1; PNZCE_neg_speech1_];
 PNZCE_pos_speech2 = [PNZCE_pos_speech2; PNZCE_pos_speech2_];
 PNZCE_neg_speech2 = [PNZCE_neg_speech2; PNZCE_neg_speech2_];
 PNZCE_pos_speech3 = [PNZCE_pos_speech3; PNZCE_pos_speech3_];
 PNZCE_neg_speech3 = [PNZCE_neg_speech3; PNZCE_neg_speech3_];
 PNZCE_pos_speech4 = [PNZCE_pos_speech4; PNZCE_pos_speech4_];
 PNZCE_neg_speech4 = [PNZCE_neg_speech4; PNZCE_neg_speech4_];
 PNZCE_pos_speech5 = [PNZCE_pos_speech5; PNZCE_pos_speech5_];
 PNZCE_neg_speech5 = [PNZCE_neg_speech5; PNZCE_neg_speech5_];
 PNZCE_pos_speech6 = [PNZCE_pos_speech6; PNZCE_pos_speech6_];
 PNZCE_neg_speech6 = [PNZCE_neg_speech6; PNZCE_neg_speech6_];
 PNZCE_pos_speech7 = [PNZCE_pos_speech7; PNZCE_pos_speech7_];
 PNZCE_neg_speech7 = [PNZCE_neg_speech7; PNZCE_neg_speech7_];

  PNZCE_pos_music1 = [PNZCE_pos_music1;  PNZCE_pos_music1_ ];
  PNZCE_neg_music1 = [PNZCE_neg_music1;  PNZCE_neg_music1_ ];
  PNZCE_pos_music2 = [PNZCE_pos_music2;  PNZCE_pos_music2_ ];
  PNZCE_neg_music2 = [PNZCE_neg_music2;  PNZCE_neg_music2_ ];
  PNZCE_pos_music3 = [PNZCE_pos_music3;  PNZCE_pos_music3_ ];
  PNZCE_neg_music3 = [PNZCE_neg_music3;  PNZCE_neg_music3_ ];
  PNZCE_pos_music4 = [PNZCE_pos_music4;  PNZCE_pos_music4_ ];
  PNZCE_neg_music4 = [PNZCE_neg_music4;  PNZCE_neg_music4_ ];
  PNZCE_pos_music5 = [PNZCE_pos_music5;  PNZCE_pos_music5_ ];
  PNZCE_neg_music5 = [PNZCE_neg_music5;  PNZCE_neg_music5_ ];
  PNZCE_pos_music6 = [PNZCE_pos_music6;  PNZCE_pos_music6_ ];
  PNZCE_neg_music6 = [PNZCE_neg_music6;  PNZCE_neg_music6_ ];
  PNZCE_pos_music7 = [PNZCE_pos_music7;  PNZCE_pos_music7_ ];
  PNZCE_neg_music7 = [PNZCE_neg_music7;  PNZCE_neg_music7_ ];


% Compute the number of tracks analyzed so far for each type of audio
    A1 = length(PNZCE_pos_speech1);
    A2 = length(PNZCE_pos_music1);

analyzed_tracks = min([A1, A2]);
[num2str(analyzed_tracks),' tracks completed .. ']

end
```

Listing A20 (Continued ...)

```
PNZCE_pos_speech1 = PNZCE_pos_speech1(1:target_tracks);
PNZCE_neg_speech1 = PNZCE_neg_speech1(1:target_tracks);
 PNZCE_pos_music1 = PNZCE_pos_music1(1:target_tracks);
 PNZCE_neg_music1 = PNZCE_neg_music1(1:target_tracks);

PNZCE_pos_speech2 = PNZCE_pos_speech2(1:target_tracks);
PNZCE_neg_speech2 = PNZCE_neg_speech2(1:target_tracks);
 PNZCE_pos_music2 = PNZCE_pos_music2(1:target_tracks);
 PNZCE_neg_music2 = PNZCE_neg_music2(1:target_tracks);

PNZCE_pos_speech3 = PNZCE_pos_speech3(1:target_tracks);
PNZCE_neg_speech3 = PNZCE_neg_speech3(1:target_tracks);
 PNZCE_pos_music3 = PNZCE_pos_music3(1:target_tracks);
 PNZCE_neg_music3 = PNZCE_neg_music3(1:target_tracks);

PNZCE_pos_speech4 = PNZCE_pos_speech4(1:target_tracks);
PNZCE_neg_speech4 = PNZCE_neg_speech4(1:target_tracks);
 PNZCE_pos_music4 = PNZCE_pos_music4(1:target_tracks);
 PNZCE_neg_music4 = PNZCE_neg_music4(1:target_tracks);

PNZCE_pos_speech5 = PNZCE_pos_speech5(1:target_tracks);
PNZCE_neg_speech5 = PNZCE_neg_speech5(1:target_tracks);
 PNZCE_pos_music5 = PNZCE_pos_music5(1:target_tracks);
 PNZCE_neg_music5 = PNZCE_neg_music5(1:target_tracks);

PNZCE_pos_speech6 = PNZCE_pos_speech6(1:target_tracks);
PNZCE_neg_speech6 = PNZCE_neg_speech6(1:target_tracks);
 PNZCE_pos_music6 = PNZCE_pos_music6(1:target_tracks);
 PNZCE_neg_music6 = PNZCE_neg_music6(1:target_tracks);

PNZCE_pos_speech7 = PNZCE_pos_speech7(1:target_tracks);
PNZCE_neg_speech7 = PNZCE_neg_speech7(1:target_tracks);
 PNZCE_pos_music7 = PNZCE_pos_music7(1:target_tracks);
 PNZCE_neg_music7 = PNZCE_neg_music7(1:target_tracks);


save PNZCE_downsampling

%% %%%%%%%%%%%%%%%%%%%%%%%%% plot P(NZCE) (event + +) %%%%%%%%%%%%%%%%%%%%
Bins = 0:0.0125:1;

% Sampling at 44.1 kHz

[ Events_speech1_Y, Events_speech1_X ] = rhist(PNZCE_pos_speech1, Bins);
[ Events_music1_Y,  Events_music1_X  ] = rhist(PNZCE_pos_music1,  Bins);

Events_speech1_Y = 100 * Events_speech1_Y;
 Events_music1_Y = 100 * Events_music1_Y;

% Sampling at 22.05 kHz

[ Events_speech2_Y, Events_speech2_X ] = rhist(PNZCE_pos_speech2, Bins);
[ Events_music2_Y,  Events_music2_X  ] = rhist(PNZCE_pos_music2,  Bins);
```

Listing A20 (Continued ...)

```
Events_speech2_Y = 100 * Events_speech2_Y;
 Events_music2_Y = 100 * Events_music2_Y;

% Sampling at 16 kHz

[ Events_speech3_Y, Events_speech3_X ] = rhist(PNZCE_pos_speech3, Bins);
[ Events_music3_Y,  Events_music3_X  ] = rhist(PNZCE_pos_music3,  Bins);

Events_speech3_Y = 100 * Events_speech3_Y;
 Events_music3_Y = 100 * Events_music3_Y;

% Sampling at 8 kHz

[ Events_speech4_Y, Events_speech4_X ] = rhist(PNZCE_pos_speech4, Bins);
[ Events_music4_Y,  Events_music4_X  ] = rhist(PNZCE_pos_music4,  Bins);

Events_speech4_Y = 100 * Events_speech4_Y;
 Events_music4_Y = 100 * Events_music4_Y;

% Sampling at 4 kHz

[ Events_speech5_Y, Events_speech5_X ] = rhist(PNZCE_pos_speech5, Bins);
[ Events_music5_Y,  Events_music5_X  ] = rhist(PNZCE_pos_music5,  Bins);

Events_speech5_Y = 100 * Events_speech5_Y;
 Events_music5_Y = 100 * Events_music5_Y;

% Sampling at 2 kHz

[ Events_speech6_Y, Events_speech6_X ] = rhist(PNZCE_pos_speech6, Bins);
[ Events_music6_Y,  Events_music6_X  ] = rhist(PNZCE_pos_music6,  Bins);

Events_speech6_Y = 100 * Events_speech6_Y;
 Events_music6_Y = 100 * Events_music6_Y;

% Sampling at 1 kHz

[ Events_speech7_Y, Events_speech7_X ] = rhist(PNZCE_pos_speech7, Bins);
[ Events_music7_Y,  Events_music7_X  ] = rhist(PNZCE_pos_music7,  Bins);

Events_speech7_Y = 100 * Events_speech7_Y;
 Events_music7_Y = 100 * Events_music7_Y;


% Detect speech in a speech/music database

[FP1,TP1,Distance1,opt_TP1,opt_FP1,opt_thr1] = ...
                                    roc(Events_speech1_Y, Events_music1_Y);

[FP2,TP2,Distance2,opt_TP2,opt_FP2,opt_thr2] = ...
                                    roc(Events_speech2_Y, Events_music2_Y);
```

Listing A20 (Continued ...)

```
[FP3,TP3,Distance3,opt_TP3,opt_FP3,opt_thr3] = ...
                                    roc(Events_speech3_Y, Events_music3_Y);

[FP4,TP4,Distance4,opt_TP4,opt_FP4,opt_thr4] = ...
                                    roc(Events_speech4_Y, Events_music4_Y);

[FP5,TP5,Distance5,opt_TP5,opt_FP5,opt_thr5] = ...
                                    roc(Events_speech5_Y, Events_music5_Y);

[FP6,TP6,Distance6,opt_TP6,opt_FP6,opt_thr6] = ...
                                    roc(Events_speech6_Y, Events_music6_Y);

[FP7,TP7,Distance7,opt_TP7,opt_FP7,opt_thr7] = ...
                                    roc(Events_speech7_Y, Events_music7_Y);

figure('color','white'), hold on
plot(FP1,TP1, 'k',   'LineWidth', 3); plot(FP2,TP2, 'r',   'LineWidth', 3)
plot(FP3,TP3, 'g',   'LineWidth', 3); plot(FP4,TP4, 'b',   'LineWidth', 3)
plot(FP5,TP5, 'r--', 'LineWidth', 3); plot(FP6,TP6, 'g--', 'LineWidth', 3)
plot(FP7,TP7, 'b--', 'LineWidth', 3)
axis([-0.01 1.01 -0.01 1.01])

title('ROC Curves Using P(NZCE_+ _+) to Detect Speech')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('44.1 kHz','22.05 kHz','16 kHz','8 kHz','4 kHz','2 kHz','1 kHz')

sampling = [fs1; fs2; fs3; fs4; fs5; fs6; fs7]/1000;
Distance = [Distance1; Distance2; Distance3; ...
            Distance4; Distance5; Distance6; Distance7];

  opt_TP = [opt_TP1; opt_TP2; opt_TP3; opt_TP4; opt_TP5; opt_TP6; opt_TP7];
  opt_FP = [opt_FP1; opt_FP2; opt_FP3; opt_FP4; opt_FP5; opt_FP6; opt_FP7];

figure('color','white'), hold on
plot(sampling, Distance, 'b', 'LineWidth', 3)

figure('color','white'), hold on
plot(sampling, opt_TP, 'b',   'LineWidth', 3)
plot(sampling, opt_FP, '--b', 'LineWidth', 3)

% Detect music in a speech/music database

[FP1,TP1,Distance1,opt_TP1,opt_FP1,opt_thr1] = ...
                                    roc2(Events_music1_Y,
Events_speech1_Y);

[FP2,TP2,Distance2,opt_TP2,opt_FP2,opt_thr2] = ...
                                    roc2(Events_music2_Y,
Events_speech2_Y);

[FP3,TP3,Distance3,opt_TP3,opt_FP3,opt_thr3] = ...
                                    roc2(Events_music3_Y,
Events_speech3_Y);
```

Listing A20 (Continued ...)

```
[FP4,TP4,Distance4,opt_TP4,opt_FP4,opt_thr4] = ...
                                    roc2(Events_music4_Y,
Events_speech4_Y);

[FP5,TP5,Distance5,opt_TP5,opt_FP5,opt_thr5] = ...
                                    roc2(Events_music5_Y,
Events_speech5_Y);

[FP6,TP6,Distance6,opt_TP6,opt_FP6,opt_thr6] = ...
                                    roc2(Events_music6_Y,
Events_speech6_Y);

[FP7,TP7,Distance7,opt_TP7,opt_FP7,opt_thr7] = ...
                                    roc2(Events_music7_Y,
Events_speech7_Y);

figure('color','white'), hold on
plot(FP1,TP1, 'k',   'LineWidth', 3); plot(FP2,TP2, 'r',   'LineWidth', 3)
plot(FP3,TP3, 'g',   'LineWidth', 3); plot(FP4,TP4, 'b',   'LineWidth', 3)
plot(FP5,TP5, 'r--', 'LineWidth', 3); plot(FP6,TP6, 'g--', 'LineWidth', 3)
plot(FP7,TP7, 'b--', 'LineWidth', 3)
axis([-0.01 1.01 -0.01 1.01])

title('ROC Curves Using P(NZCE_+ _+) to Detect Music')
xlabel('False Positive Rate (FP)')
ylabel('True Positive Rate (TP)')
legend('44.1 kHz','22.05 kHz','16 kHz','8 kHz','4 kHz','2 kHz','1 kHz')

Distance = [Distance1; Distance2; Distance3; ...
            Distance4; Distance5; Distance6; Distance7];

  opt_TP = [opt_TP1; opt_TP2; opt_TP3; opt_TP4; opt_TP5; opt_TP6; opt_TP7];
  opt_FP = [opt_FP1; opt_FP2; opt_FP3; opt_FP4; opt_FP5; opt_FP6; opt_FP7];

figure('color','white'), hold on
plot(sampling, Distance, 'r', 'LineWidth', 3)

figure(3)
plot(sampling, opt_TP, 'r',   'LineWidth', 3)
plot(sampling, opt_FP, '--r', 'LineWidth', 3)
title('Optimum TP/FP Rates from P(NZCE_+ _+) ROC Curves')
xlabel('Sampling Frequency, kHz')
ylabel('TP/FP Rates')
legend('Speech TP Rate','Speech FP Rate','Music TP Rate','Music FP Rate')
xlim([0 44.1])
```

Listing A20 (Continued ...)

```
%% %%%%%%%%%%%%%%%%%%%%%%%% plot P(NZCE) (event - -) %%%%%%%%%%%%%%%%%%%

Bins = 0:0.0125:1;

% Sampling at 44.1 kHz

[ Events_speech1_Y, Events_speech1_X ] = rhist(PNZCE_neg_speech1, Bins);
[ Events_music1_Y,  Events_music1_X  ] = rhist(PNZCE_neg_music1,  Bins);

Events_speech1_Y = 100 * Events_speech1_Y;
 Events_music1_Y = 100 * Events_music1_Y;

% Sampling at 22.05 kHz

[ Events_speech2_Y, Events_speech2_X ] = rhist(PNZCE_neg_speech2, Bins);
[ Events_music2_Y,  Events_music2_X  ] = rhist(PNZCE_neg_music2,  Bins);

Events_speech2_Y = 100 * Events_speech2_Y;
 Events_music2_Y = 100 * Events_music2_Y;

% Sampling at 16 kHz

[ Events_speech3_Y, Events_speech3_X ] = rhist(PNZCE_neg_speech3, Bins);
[ Events_music3_Y,  Events_music3_X  ] = rhist(PNZCE_neg_music3,  Bins);

Events_speech3_Y = 100 * Events_speech3_Y;
 Events_music3_Y = 100 * Events_music3_Y;

% Sampling at 8 kHz

[ Events_speech4_Y, Events_speech4_X ] = rhist(PNZCE_neg_speech4, Bins);
[ Events_music4_Y,  Events_music4_X  ] = rhist(PNZCE_neg_music4,  Bins);

Events_speech4_Y = 100 * Events_speech4_Y;
 Events_music4_Y = 100 * Events_music4_Y;

% Sampling at 4 kHz

[ Events_speech5_Y, Events_speech5_X ] = rhist(PNZCE_neg_speech5, Bins);
[ Events_music5_Y,  Events_music5_X  ] = rhist(PNZCE_neg_music5,  Bins);

Events_speech5_Y = 100 * Events_speech5_Y;
 Events_music5_Y = 100 * Events_music5_Y;

% Sampling at 2 kHz

[ Events_speech6_Y, Events_speech6_X ] = rhist(PNZCE_neg_speech6, Bins);
[ Events_music6_Y,  Events_music6_X  ] = rhist(PNZCE_neg_music6,  Bins);

Events_speech6_Y = 100 * Events_speech6_Y;
 Events_music6_Y = 100 * Events_music6_Y;
```

Listing A20 (Continued ...)

```
% Sampling at 1 kHz

[ Events_speech7_Y, Events_speech7_X ] = rhist(PNZCE_neg_speech7, Bins);
[ Events_music7_Y,  Events_music7_X  ] = rhist(PNZCE_neg_music7,  Bins);

Events_speech7_Y = 100 * Events_speech7_Y;
 Events_music7_Y = 100 * Events_music7_Y;

% Detect speech in a speech/music database

[FP1,TP1,Distance1,opt_TP1,opt_FP1,opt_thr1] = ...
                                    roc2(Events_speech1_Y, Events_music1_Y);

[FP2,TP2,Distance2,opt_TP2,opt_FP2,opt_thr2] = ...
                                    roc2(Events_speech2_Y, Events_music2_Y);

[FP3,TP3,Distance3,opt_TP3,opt_FP3,opt_thr3] = ...
                                    roc2(Events_speech3_Y, Events_music3_Y);

[FP4,TP4,Distance4,opt_TP4,opt_FP4,opt_thr4] = ...
                                    roc2(Events_speech4_Y, Events_music4_Y);

[FP5,TP5,Distance5,opt_TP5,opt_FP5,opt_thr5] = ...
                                    roc2(Events_speech5_Y, Events_music5_Y);

[FP6,TP6,Distance6,opt_TP6,opt_FP6,opt_thr6] = ...
                                    roc2(Events_speech6_Y, Events_music6_Y);

[FP7,TP7,Distance7,opt_TP7,opt_FP7,opt_thr7] = ...
                                    roc2(Events_speech7_Y, Events_music7_Y);

figure('color','white'), hold on
plot(FP1,TP1, 'k',   'LineWidth', 3); plot(FP2,TP2, 'r',   'LineWidth', 3)
plot(FP3,TP3, 'g',   'LineWidth', 3); plot(FP4,TP4, 'b',   'LineWidth', 3)
plot(FP5,TP5, 'r--', 'LineWidth', 3); plot(FP6,TP6, 'g--', 'LineWidth', 3)
plot(FP7,TP7, 'b--', 'LineWidth', 3)
axis([-0.01 1.01 -0.01 1.01])

title('ROC Curves Using P(NZCE_- _-) to Detect Speech')
xlabel('False Negitive Rate (FP)')
ylabel('True Negitive Rate (TP)')
legend('44.1 kHz','22.05 kHz','16 kHz','8 kHz','4 kHz','2 kHz','1 kHz')

sampling = [fs1; fs2; fs3; fs4; fs5; fs6; fs7]/1000;
Distance = [Distance1; Distance2; Distance3; ...
            Distance4; Distance5; Distance6; Distance7];

  opt_TP = [opt_TP1; opt_TP2; opt_TP3; opt_TP4; opt_TP5; opt_TP6; opt_TP7];
  opt_FP = [opt_FP1; opt_FP2; opt_FP3; opt_FP4; opt_FP5; opt_FP6; opt_FP7];

figure(2)
plot(sampling, Distance, 'b--', 'LineWidth', 3)
title('Euclidian Distance for Speech ROC Curves')
xlabel('Sampling Frequency, kHz'); ylabel('Euclidian Distance')
legend('P(NZCE_+ _+)', 'P(NZCE_- _-)')
```

Listing A20 (Continued ...)

```
figure('color','white'), hold on
plot(sampling, opt_TP, 'b',   'LineWidth', 3)
plot(sampling, opt_FP, '--b', 'LineWidth', 3)

% Detect music in a speech/music database
[FP1,TP1,Distance1,opt_TP1,opt_FP1,opt_thr1] = ...
                                    roc(Events_music1_Y, Events_speech1_Y);

[FP2,TP2,Distance2,opt_TP2,opt_FP2,opt_thr2] = ...
                                    roc(Events_music2_Y, Events_speech2_Y);

[FP3,TP3,Distance3,opt_TP3,opt_FP3,opt_thr3] = ...
                                    roc(Events_music3_Y, Events_speech3_Y);

[FP4,TP4,Distance4,opt_TP4,opt_FP4,opt_thr4] = ...
                                    roc(Events_music4_Y, Events_speech4_Y);

[FP5,TP5,Distance5,opt_TP5,opt_FP5,opt_thr5] = ...
                                    roc(Events_music5_Y, Events_speech5_Y);

[FP6,TP6,Distance6,opt_TP6,opt_FP6,opt_thr6] = ...
                                    roc(Events_music6_Y, Events_speech6_Y);

[FP7,TP7,Distance7,opt_TP7,opt_FP7,opt_thr7] = ...
                                    roc(Events_music7_Y, Events_speech7_Y);

figure('color','white'), hold on
plot(FP1,TP1, 'k',   'LineWidth', 3); plot(FP2,TP2, 'r',   'LineWidth', 3)
plot(FP3,TP3, 'g',   'LineWidth', 3); plot(FP4,TP4, 'b',   'LineWidth', 3)
plot(FP5,TP5, 'r--', 'LineWidth', 3); plot(FP6,TP6, 'g--', 'LineWidth', 3)
plot(FP7,TP7, 'b--', 'LineWidth', 3)
axis([-0.01 1.01 -0.01 1.01])

title('ROC Curves Using P(NZCE_- _-) to Detect Music')
xlabel('False Negitive Rate (FP)')
ylabel('True Negitive Rate (TP)')
legend('44.1 kHz','22.05 kHz','16 kHz','8 kHz','4 kHz','2 kHz','1 kHz')

Distance = [Distance1; Distance2; Distance3; ...
            Distance4; Distance5; Distance6; Distance7];

  opt_TP = [opt_TP1; opt_TP2; opt_TP3; opt_TP4; opt_TP5; opt_TP6; opt_TP7];
  opt_FP = [opt_FP1; opt_FP2; opt_FP3; opt_FP4; opt_FP5; opt_FP6; opt_FP7];

figure(5)
plot(sampling, Distance, 'r--', 'LineWidth', 3)
title('Euclidian Distance for Music ROC Curves')
xlabel('Sampling Frequency, kHz')
ylabel('Euclidian Distance')
legend('P(NZCE_+ _+)', 'P(NZCE_- _-)')
```

Listing A20 (Continued ...)

```
figure(7)
plot(sampling, opt_TP, 'r',   'LineWidth', 3)
plot(sampling, opt_FP, '--r', 'LineWidth', 3)
title('Optimum TP/FP Rates from P(NZCE_- _-) ROC Curves')
xlabel('Sampling Frequency, kHz')
ylabel('TP/FP Rates')
legend('Speech TP Rate','Speech FP Rate','Music TP Rate','Music FP Rate')
xlim([0 44.1])
```

Appendix B: Publications

The following two papers were published in two IEEE conferences. They are the outcome of chapter 4. Both papers can be obtained from ***IEEE Xplore*** website as shown in figure B.1. A third paper is submitted to IET for publication and is under review.

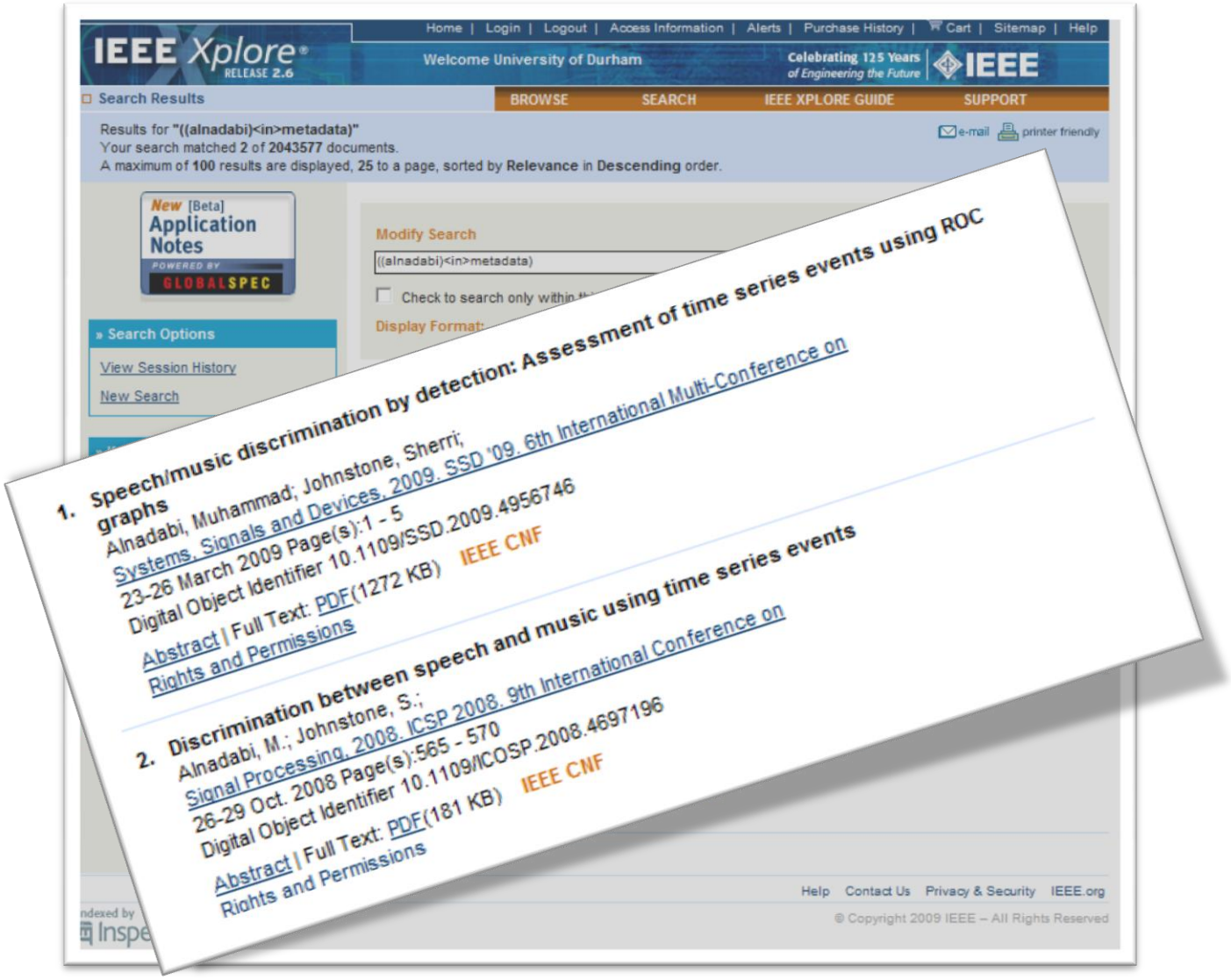


Figure B.1: IEEE publications in two conferences: ICSP 2008 in Beijing, (China) and SSD'09 in Djerba (Tunisia)